UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

Análisis del algoritmo Solovay-Kitaev para aproximación de puertas cuánticas

Sara Sánchez Martín

Tutor: Gonzalo Martínez Muñoz

JUNIO 2019

Análisis del algoritmo Solovay-Kitaev para aproximación de puertas cuánticas

AUTOR: Sara Sánchez Martín
TUTOR: Gonzalo Martínez Muñoz

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2019

RESUMEN

Este Trabajo Fin de Grado tiene como objetivo el estudio y análisis de un algoritmo de aproximación de puertas cuánticas llamado algoritmo de Solovay-Kitaev.

El paradigma de la computación cuántica se encuentra aún en fase de desarrollo. Los inicios se remontan a los postulados desarrollados a principios del siglo XX por físicos como Schrödinger, Dirac, y Von Neumann, entre otros, pero no ha sido hasta los años 2000 cuando se han logrado sintetizar los primeros *qbits* y presentar los primeros ordenadores cuánticos comerciales como la desarrollada por D-Wave System en 2013.

La unidad de información de un ordenador cuántico es un *qbit*. Los *qbits* tienen la particularidad de que no almacenan información discreta que sólo pueda ser 0 o 1, como un bit, sino que representan ambos al mismo tiempo, cada uno con un valor de probabilidad. Esto se conoce como dualidad, la cual desaparece al medir el valor del *qbit*, esto es, que colapsa irreversiblemente en 0 o en 1.

Los sistemas basados en N *qbits* se identifican con elementos de un espacio de Hilbert de dimensión 2^N. Son vectores de 2^N coordenadas, donde cada una es un número complejo. El cuadrado de la norma de cada coeficiente determina la probabilidad de que el *qbit* colapse en cada estado posible. El espacio donde puede vivir un *qbit* de D dimensiones se identifica con la superficie de una hiperesfera D-dimensional, llamada esfera de Bloch cuando D=2, o espacio simétrico en otro caso.

A diferencia de las puertas lógicas para bits, las puertas cuánticas son reversibles. Esto es gracias a que, como un *qbit* es análogo a un vector, una puerta cuántica es análoga a una matriz, y no a una cualquiera, sino que la naturaleza de los *qbits* fuerza a que se traten de matrices unitarias (con coeficientes complejos), las cuales son invertibles.

El teorema de Solovay-Kitaev afirma que dado un conjunto inicial bien escogido de puertas, se puede encontrar una buena aproximación a cualquier puerta deseada por medio de una combinación de puertas de este conjunto original. En el desarrollo del código *Python* del algoritmo (para un único *qbit*), el conjunto original de puertas es un subgrupo denso del grupo unitario especial $\mathbb{SU}(2)$. Se trata de un algoritmo recursivo que en cada nivel obtiene mejores aproximaciones operando con la matriz que se desea aproximar.

Las pruebas realizadas confirman que el algoritmo presenta una complejidad de potencia logarítmica inversamente proporcional al error, $O(\ln^{2.71}(1/\varepsilon))$, y proporciona una aproximación de una longitud de cadena de $O(\ln^{3.97}(1/\varepsilon))$. Además se ha logrado completar un método de búsqueda basado en árbol-KD que ha permitido obtener la misma eficacia que la búsqueda lineal en un tiempo muchísimo menor.

El algoritmo ha demostrado ser un buen candidato en la obtención de un grupo de puertas cuánticas universales. Aunque requiere un estudio de sus limitaciones, la implementación para sistemas de N *qbits* y nuevos métodos de elección de subgrupos puede ofrecer resultados muy prometedores.

ASTRACT

The aim of this Bachelor Thesis is to study and analyze an algorithm called the Solovay Kitaev algorithm to accurately approximate quantum gates.

Quantum computing is a paradigm that is still in a development phase. Quantum physics principles date back to the early 20th century and they were developed by physicists like Schrödinger, Dirac and Von Neumann, among others. But it was not until the 2000s when it has been successfully synthesized the first *qbits* and has been present the first commercial quantum computers as the one developed by D-Wave System in 2013.

The basic information unit of a quantum computer is a *qbit*. The *qbits* do not store discrete information such 0 or 1, as a bit, but represent both at the same time, each with a probability value. This is known as duality, which disappears as the value of the *qbit* is measured, that is, that collapses irreversibly into 0 or 1.

The N *qbits* systems are identified with elements of a Hilbert space of dimension 2^N. They are vectors of 2^N coordinates, where each one is a complex number. The probability of the *qbit* to collapse into each possible state is given by the square of the norm of each coefficient. The space where a *qbit* of D dimensions can live is identified with the surface of a D-dimensional hypersphere, called the Bloch Sphere when D = 2, or Symmetric Space otherwise.

A *qbit* is comparable to a vector, and due to its nature, a quantum gate is analogous to a matrix that is forced to be unitary and with complex coefficients. Since unitary matrix are invertible, the modifications done by quantum gates can be reversed.

The theorem of Solovay Kitaev states that given a well-chosen initial set of gates, a good approximation to any desired gate can be found from of a combination of gates of the original set. The algorithm's *Python* code implementation (for a single *qbit*) is based on a dense subgroup of the Special Unitary Group $\mathbb{SU}(2)$ as original set. It is a recursive algorithm that in each level obtains better approximations operating with the matrix to approximate.

The carried out tests confirm that the algorithm has a complexity with an asymptotic tendency of $O(ln^{2.71}(1/\varepsilon))$, where ε is the reached tolerance, and provides an approximation with a chain length of $O(ln^{3.97}(1/\varepsilon))$. It has also been completed a search method based on a KD-tree thanks to that has been possible to obtain the same error as the linear search in a much better time.

The algorithm has proved to be a good candidate in obtaining a group of universal quantum gates. Although it requires an analysis of its limitations, the implementation for systems of N *qbits* and new methods of choosing good subgroups can offer very promising results.

PALABRAS CLAVE

Computación cuántica, complejidad algorítmica, tendencia asintótica, algoritmo recursivo, subgrupo, cúbit, vector, puerta cuántica, matriz, aproximación.

KEYWORDS

Quantum computing, algorithmic complexity, asymptotic tendency, recursive algorithm, subgroup, *qbit*, vector, quantum gate, matrix, approximation.

AGRADECIMIENTOS

Al Dr. Paul Pham, por su estupendo desarrollo de un repositorio público que ha permitido la implementación y desarrollo de este trabajo, y por su ayuda ofrecida en los inicios del análisis del algoritmo.

A mis tutores Gonzalo Martínez y Estrella Pulido, por haberme guiado desde que me interesé por este proyecto hace ya más de un año, su disponibilidad y sus consejos e ideas para hacer este trabajo más ameno.

A mi tutor de grado Alfonso Ortega, por su interés por mi bienestar durante toda mi carrera, por abrirme la puerta al mundo laboral y por ser una persona con la que siempre me alegro de cruzarme en los pasillos.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Desde el nacimiento de las primeras sociedades, las personas siempre han estado invadidas por la necesidad de buscar una explicación a todo lo que les rodea. Esta necesidad se cubrió con todo tipo de mitos y leyendas hasta que nació la ciencia. La física es la rama de la ciencia que estudia la mecánica y el comportamiento del mundo, así como las interacciones entre sus componentes, materia y energía.

Como la mayoría de las ciencias, a lo largo de la historia la física ha sido cuestionada por sectores de la sociedad que se mostraban escépticos ante los descubrimientos científicos que iban adquiriendo forma. Pero lejos de rendirse, a medida que mejoraban los recursos, los físicos continuaron compartiendo sus teorías y contribuyendo en la búsqueda de la verdad absoluta.

El interés por este trabajo de fin de grado debe su origen, en parte, al interés compartido con los físicos de la historia por entender y dar forma al mundo que nos rodea. Pero sobre todo, al hecho de poder participar activamente en el proceso de sacar adelante y dar fuerza al paradigma de la computación cuántica, una nueva rama que combina la física con la computación y cuyas expectativas resultan tan prometedoras como interesantes.

1.2 Objetivos

Originalmente, este trabajo de fin de grado se pensó para tratar a cerca de algoritmos cuánticos y centrar su interés en el algoritmo de Grover. Pero durante las primeras investigaciones y lecturas, apareció un artículo muy interesante sobre un algoritmo cuántico llamado Solovay-Kitaev que, apoyado por un teorema con el mismo nombre, asegura que puede aproximar con una precisión ε cualquier puerta cuántica en un tiempo polilogarítmico en $1/\varepsilon$.

El objetivo de este trabajo es estudiar y analizar el algoritmo de Solvay-Kitaev. Para ello, en primer lugar se realiza una parada por las bases de la computación y física cuántica, así como toda la teoría matemática en la que se apoya. Esta memoria hace hincapié en las estructuras algebraicas básicas y los fundamentos de álgebra matricial y espacios vectoriales, conceptos matemáticos fundamentales para poder axiomatizar y formalizar la gran mayoría de las ciencias puras y, en espacial, la naturaleza de los elementos de la computación cuántica.

Para el análisis del algoritmo se establecen ciertos parámetros variables, como la profundidad de la recursión, el tamaño del conjunto original de puertas, y la forma de buscar la mejor aproximación para el caso base. El objetivo del estudio es determinar si la variación y combinación de estas propiedades ajustables afecta al rendimiento del algoritmo positiva o negativamente, en mayor o menor medida y, por último, se pretende comprobar empíricamente si la tendencia asintótica coincide con lo que estipula el teorema de Solovay-Kitaev.

1.3 Organización de la memoria

La memoria está comprendida por los siguientes capítulos:

- **Estado del arte:** Se describe el proceso por el cual ha pasado es estudio de la física cuántica a lo largo de su historia, desde los primeros experimentos y postulados hasta la creación del primer *qbit* y ordenadores cuánticos en el siglo XXI.

- **Conceptos básicos y análisis del algoritmo:** Este capítulo comienza haciendo una parada por los conceptos de álgebra y cálculo en los que se basa la formalización de un sistema cuántico. Más tarde analiza hasta qué punto están relacionados y finaliza con un estudio detallado del teorema y algoritmo de Solovay-Kitaev.

- **Implementación del algoritmo:** Se describen los módulos, librerías y capas que componen el código del algoritmo, desde las operaciones a más bajo hasta la representación de gráficas de los datos.

- **Resultados y validación del teorema:** En este capítulo se exponen y analizan una serie de gráficas que recogen resultados sobre la eficacia y eficiencia del algoritmo para después comprobar y valorar el teorema de Solovay-Kitaev

- **Conclusiones y trabajo futuro:** El contenido de la memoria concluye en este capítulo. Ofrece una perspectiva más personal sobre el paradigma de la cuántica y todo lo que le rodea y propone algunas mejoras futuras interesantes para complementar este trabajo.

Esta memoria consta también de seis anexos cuyo objetivo es complementar el contenido y añadir información sobre algunos conceptos en segundo plano que no formaban parte del desarrollo principal pero merecían especial atención.

2 Estado del arte

Este capítulo recoge los sucesos que provocan y acompañan al auge de la física cuántica, partiendo de las limitaciones de los postulados de la teoría de la mecánica clásica. Veremos cómo los recientes descubrimientos en el campo de la física de partículas han permitido sacar provecho de los principios de superposición y entrelazamiento cuántico para poder desarrollar ordenadores con una capacidad de cómputo mucho mayor que la de los ordenadores clásicos.

No obstante, como afirma Moshe Y. Vardi en su artículo [1], ciertos escépticos dudan de la computación cuántica en cuanto a la posibilidad de poder implementarla en el mundo real. Éstos sostienen que, a pesar de que aportaría una ventaja exponencial sobre la computación clásica, no es físicamente posible construir ordenadores cuánticos a una escala manejable para el ser humano. Su argumento se basa en que los *qbits*, la unidad básica de información de un ordenador cuántico, se encuentran sometidos a inevitables interacciones con el mundo exterior al computador, lo cual altera su estado y modifica la exactitud de las mediciones. Aun así, muchos de estos escépticos sí que otorgan el beneficio de la duda a la creación de ordenadores cuánticos, convirtiendo este paradigma en una nueva y poco confiable revolución en la física.

2.1 Física cuántica

La mecánica cuántica, o simplemente cuántica, es la rama de la física que describe el funcionamiento de la naturaleza a pequeña escala.

Surgió a principios del siglo XX dentro del marco de la física tradicional, cuando las teorías conocidas hasta la fecha no permitían dar explicación a las nuevas incógnitas de esta ciencia. Pretendía explicar ciertos fenómenos como, por ejemplo, el espectro de la radiación del cuerpo negro, la dualidad onda-partícula, la desigualdad de Bell y el efecto fotoeléctrico. La cuántica proporciona el fundamento de la fenomenología del átomo, de su núcleo y de las partículas elementales. Los impactos en teoría de la información, criptografía y química han sido decisivos para prolongar su estudio.

El desarrollo formal de la teoría fue obra de los esfuerzos conjuntos de varios físicos y matemáticos de la época como Schrödinger, Heisenberg, Einstein, Dirac, Bohr y Von Neumann entre otros. Algunos de los aspectos fundamentales de la teoría están siendo aún estudiados activamente, así como sus reveladoras aplicaciones en campos tanto de física como de química. Por ejemplo, en la física del estado sólido, las leyes de la mecánica cuántica han permitido comprender mejor el comportamiento de los electrones.

Expresado matemáticamente, según los postulados desarrollados por Dirac y Von Neumann, un sistema cuántico presenta sus posibles estados en forma de vectores unitarios pertenecientes a un espacio de Hilbert complejo separable, llamado espacio de estados, el cual viene determinado por las características del sistema. Un espacio de Hilbert se define como un espacio de producto interior que es completo con respecto a la norma vectorial definida por dicho producto interior. Complejo hace referencia a que se define sobre el cuerpo de números complejos $\mathbb{C}$, y separable a que admite una base ortonormal numerable. Más detalles acerca de espacios vectoriales y espacios de Hilbert se ven en la sección 3.2. La evolución temporal de un estado de este espacio viene descrita por la ecuación de Schrödinger, detallada en el Anexo A.

Originalmente, la teoría cuántica entraba en contradicción con los postulados de la mecánica clásica, y en particular con el principio de localidad defendido por Einstein, quien calificó la teoría cuántica como incompleta. Como justificación de este argumento surgió la paradoja de Eintein-Podolsky, según la cual

el hecho de medir una partícula puede alterar el estado de su análogo enlazado ([3]). Esto es una consecuencia del principio del entrelazamiento cuántico.

El término de entrelazamiento cuántico fue introducido por Erwin Schrödinger en 1935 ([3]). Schrödinger postula que un conjunto de partículas entrelazadas comparten una única función de onda (ver Anexo A) en un mismo sistema en lugar de comportarse como elementos independientes, es decir, que sus estados físicos se encuentran "ligados". La tarea de cuantificar (medir) el entrelazamiento encontrando una magnitud no es trivial. Como se trata de una magnitud binaria (dos partículas o están entrelazas o no), esta misión se reduce a conseguir detectar el entrelazamiento. Se encuentra aún en proceso, aunque existen ciertos sistemas que sí se han logrado establecer como o bien máximamente entrelazados o lo contrario, como se verá en capítulos posteriores.

El entrelazamiento de partículas es un fenómeno esencial para desarrollar ordenadores cuánticos para poder alcanzar las mejores velocidades de computación que prometen. Esto es gracias al hecho de que dos partículas entrelazadas comparten información, por lo que observar y determinar el valor de una de ellas permite instantáneamente conocer el valor de la otra aunque se encuentren a grandes distancias.

2.2 Computación cuántica

La principal razón que condujo al progreso científico a estudiar más a fondo y aprovechar los fenómenos de la física anteriormente mencionados parte de la necesidad de controlar el comportamiento de las partículas subatómicas que operan dentro de los transistores. Estas partículas viven en espacios muy pequeños y, además, estas escalas disminuyen muy rápidamente a medida que pasa el tiempo y las necesidades tecnológicas aumentan. Este fenómeno se conoce como efecto túnel, que toma lugar cuando una partícula, y en concreto un electrón, alcanza un nivel de potencial o impedancia mayor que su propia energía cinética. Al tener comportamiento de onda, el electrón "escapa" del espacio controlado dentro del cual debería operar teóricamente, produciéndose un comportamiento incorrecto de los microchips a los que sirven e imposibilitando su control: es en este momento cuando entran en juego las leyes de la física cuántica.

Ya fue a finales de la década de los años 70 cuando Paul Benioff expuso sus ideas para trabajar a nivel de cuantos en la computación, y no a nivel de voltaje eléctrico ([4]). Un cuanto se define como la cantidad discreta más pequeña de energía que puede ser absorbida, propagada o emitida por la materia. Mientras que, en computación clásica, la unidad mínima de información, un bit, puede tomar sólo dos valores discretos (0 o 1), un bit cuántico puede tomar "al mismo tiempo" infinitos valores entre 0 y 1. Es lo que se conoce como superposición. Esto permite un número de operaciones simultáneas exponencial al número de unidades de información. Es así como un bit pasa a denominarse *qbit*, o *cúbit*. El uso de *qbits* en computación permite aumentar considerablemente el rendimiento de un procesador: mientras que actualmente la supercomputadora Summit ([2]) tiene capacidad para soportar 200 *petaflops* (FLOPS, del inglés "floating point operations per second"), un ordenador de sólo 30 *qbits* equivaldría a un procesador convencional de 10 *teraflops*.

No obstante, este nuevo paradigma trae consigo varios problemas, entre ellos, la decoherencia cuántica, y la escalabilidad. La decoherencia cuántica es el resultado del "desentrelazamiento", es decir, el fenómeno por el cual las partículas entrelazadas pierden su función de onda compartida. Esto provoca que las operaciones deban ser realizadas en un tiempo inferior al tiempo de decoherencia para así minimizar la tasa de error. Se cita con frecuencia una tasa de error límite de 10^{-4}, por debajo de la cual se supone que sería posible la aplicación eficaz de la corrección de errores cuánticos. Por otro lado, a medida que aumenta la complejidad de un problema, aumenta el número de *qbits* necesarios para su implementación, y con él, la tasa media de errores. A más tasa de errores, mayor es también el número de *qbits* necesarios para su corrección. Por todo ello, la realización de un diseño capaz de llevar a cabo cálculos arbitrariamente largos sin alterar su precisión no es nada trivial de conseguir.

Un reciente artículo ([5]) sostiene ideas poco alentadoras sobre el auge del campo de la cuántica. Éste afirma que la implementación empírica de sistemas cuánticos llegaría a causar tal revolución que no sería posible adaptar todos los sistemas modernos, tales como los protocolos de red, a esta nueva teoría en un tiempo menor de una década. Además, este cambio revolucionario debería lidiar con nuevos peligros, ya que el uso de algoritmos cuánticos puede comprometer los sistemas de cifrado de clave pública, que son la base de la seguridad de la gran mayoría de redes y computadores modernos.

Cabe destacar algunos de los algoritmos cuánticos que han logrado ser propuestos y desarrollados logrando unas tasas de error comparables a las de los ordenadores clásicos, estos son el Algoritmo de Shor, Algoritmo de Grover y Algoritmo de Deutsch-Jozsa, que serán comentados más adelante.

A finales de la década de los 90, investigadores de Los Álamos y el Instituto Tecnológico de Massachusetts consiguieron canalizar a través de una solución de aminoácidos el primer *qbit*. Esto permitió, años más tarde, ejecutar el algoritmo de búsqueda de Grover en una máquina de 3 *qbits*.

Se estima que durante los próximos 15 años se puedan empezar a comercializar los primeros ordenadores cuánticos, similares al presentado por IBM ([6]), el cual cuenta con un procesador de 17 *qbits*. También cabe destacar el trabajo de D-Wave System, empresa que lanzó en 2013 un computador con un poder de cálculo de 439 *qbits* tras 10 años de investigación. Actualmente esta empresa cuenta con un ordenador cuántico comercial de 2048 *qbits*.

2.3 Algoritmos cuánticos

Existen algunos de los algoritmos cuánticos que han logrado ser propuestos y desarrollados y han logrado alcanzar unas tasas de error comparables a las de los ordenadores clásicos. Estos son el Algoritmo de Shor, Algoritmo de Grover y Algoritmo de Deutsch-Jozsa.

David Deutsch y Richard Jozsa propusieron en 1992 el algoritmo Deutsch-Jozsa ([7]). A pesar de no tener apenas utilidad práctica, fue uno de los primeros algoritmos cuánticos verificados como exponencialmente más rápidos que cualquier algoritmo determinista. Su utilidad consiste en determinar si una función $f: \mathbb{Z}_2^n \to \mathbb{Z}_2$, es constante o, por el contrario, el bit de salida depende de los bits de entrada en una determinada proporción. Fue mejorado posteriormente por Richard Cleve, Artur Ekert, Chiara Macchiavello y Michelle Mosca en 1998 ([8]).

Más tarde, Peter Shor propuso en 1995 el algoritmo de Shor ([9]). Éste es capaz de descomponer en factores primos un número N en tiempo $O(log^3(N))$ y espacio $O(log(N))$ basando sus cálculos en aritmética modular. Este algoritmo está relacionado con el problema RSA, de vital importancia en criptografía, por lo que ha promovido una importante atención a la computación cuántica. Peter Shor propuso también en 1995 la primera corrección de errores en cuántica ([10]), lo cual supuso un importante avance en estos campos debido a la imposibilidad de usar mecanismos clásicos para generar algoritmos cuánticos a prueba de errores.

Por último, Lov Grover publicó en 1996 el algoritmo de Grover ([11]). Su objetivo es la búsqueda en bases de datos no necesariamente ordenadas. Dadas N entradas, requiere un número de pasos del orden de $O(\sqrt{N})$ y un uso de espacio de memoria de $O(log(N))$. Este algoritmo permite resolver problemas de forma más rápida que con el mejor algoritmo clásico posible, que es de orden lineal.

3 Conceptos básicos y análisis del algoritmo

Este capítulo recoge y analiza los aspectos clave de la cuántica necesarios para introducir al lector en este paradigma. Una vez establecidos, los principios cuánticos básicos y el objetivo del algoritmo del Solovay-Kitaev de aproximar cualquier puerta cuántica a partir de un conjunto finito de puertas son extrapolables a conceptos y elementos meramente algebraicos, los que también se detallan en este capítulo, pues toda la implementación y operaciones necesarias se apoyan en argumentos matemáticos.

3.1 Bases de la computación cuántica

En esta sección se introducen brevemente conceptos propios de la computación cuántica en cuanto a sus diferencias frente a la computación clásica.

3.1.1 Qbits y puertas lógicas

Como ya se comentó en el capítulo anterior, la particularidad principal de la computación cuántica es el uso de *qbits* frente a bits. El término *qbit* se le atribuye a Benjamin Schumacher, quien describe en la publicación [12] una forma de almacenar información comprimida en el número más pequeño posible de estados, lo que se conoce como compresión de Schumacher.

El valor de información contenido en un *qbit* se representa habitualmente con notación *bra-ket* ([13]), también conocida como notación de Dirac. Esta notación se utiliza además para denotar vectores abstractos y funcionales lineales en matemáticas puras. En mecánica cuántica, el estado de un sistema físico se identifica con un vector, llamado *ket*, que se denota como $|\varphi\rangle$. La aplicación del *bra* $\langle\psi|$ al *ket* $|\varphi\rangle$ da lugar a un número complejo, que se denota $\langle\psi|\varphi\rangle$, y que se corresponde con la amplitud de probabilidad para que el estado ψ colapse en el estado φ.

Un *qbit* se representa como una superposición o combinación lineal de los estados básicos $|0\rangle$ y $|1\rangle$: $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, donde las amplitudes de probabilidad α y β son números complejos: contienen información de fase, esto es, el argumento del número complejo escrito en forma polar. El Anexo C ofrece una descripción de la representación de números complejos. La probabilidad de obtener el valor 0 o 1 está determinada, respectivamente, por $|\alpha|^2$ y $|\beta|^2$. Como la probabilidad total ha de ser la unidad, necesariamente α y β están relacionados por la ecuación:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{1}$$

Cualquier medida del *qbit* altera inevitablemente su estado: la superposición se rompe y colapsa de forma irreversible en {0, 1}.

La particularidad de los *qbits* de almacenar simultáneamente valores entre 0 y 1 se denomina paralelismo cuántico. De esta forma, las operaciones de los algoritmos cuánticos que operen sobre varios estados de superposición se aplican a todas las combinaciones de entradas. Por ejemplo los dos *qbits*

$$\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \tag{2}$$

representan simultáneamente las combinaciones 00, 01, 10 y 11.

La parte derecha de (2) da lugar a un estado de dos *qbits* no entrelazados, pues pueden darse las cuatro posibilidades. Esto es posible porque los dos *qbits* de la combinación son separables (factorizables), pues la parte de la izquierda puede escribirse como un producto (ecuación (3)), al contrario que la ecuación (4), que representa a dos *qbits* en el estado de Bell.

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad (3)$$

$$|\beta_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad (4)$$

El símbolo "⊗" denota la operación de producto tensorial. Cabe destacar que cualquier producto tensorial entre estados de *qbits* da lugar a un estado conjunto de *N qbits*, como por ejemplo:

$$\frac{1}{2}(|0\rangle + \sqrt{3}|1\rangle) \otimes \frac{1}{2}(\sqrt{3}|0\rangle - |1\rangle) = \frac{1}{4}(\sqrt{3}|00\rangle - |01\rangle + 3|10\rangle - \sqrt{3}|11\rangle)$$

Esta correspondencia no es biunívoca, es decir, existen estados de *N qbits* que no pueden descomponerse como producto de los estados de los *N qbits* independientes: son los que se denominan entrelazados, como el estado de Bell de la ecuación (4).

El modelo de computación cuántica que más nos interesa es el circuito cuántico, el cual aplica puertas lógicas sobre los *qbits,* llamadas en este caso puertas cuánticas (ver Anexo B). En este modelo, los algoritmos se expresan como una serie de puertas que actúan sobre uno o varios *qbits*. Su principal disparidad frente a las puertas lógicas clásicas es que las puertas cuánticas son reversibles, mientras que las clásicas no. Por esta razón, necesariamente el número de *qbits* de entrada ha de coincidir con el número de *qbits* de salida. Dos ejemplos son la puerta de negación controlada, o CNOT (equivalente a una puerta XOR clásica), en la que el *qbit* objetivo se cambia de $|0\rangle$ a $|1\rangle$ y viceversa sí y solo sí el valor del *qbit* de control es $|1\rangle$ (Tabla 1), y la puerta Hadamard, que actúa sobre un único *qbit* (Tabla 2). Nótese que el valor de los *qbits* de control también se conserva a la salida pero, como nunca se ve alterado, no está representado en los ejemplos.

Qbit entrada	*Qbit* control	Salida
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

Tabla 1: Puerta de negación controlada / CNOT

Qbit entrada	Salida
$\|0\rangle$	$\frac{\|0\rangle + \|1\rangle}{\sqrt{2}}$
$\|1\rangle$	$\frac{\|0\rangle - \|1\rangle}{\sqrt{2}}$

Tabla 2: Puerta Hadamard

La restricción dada por la ecuación (1), junto con otras propiedades de los *qbits* que se comentan en secciones posteriores, hace que se pueda establecer una correspondencia entre la variedad donde "viven" y la superficie de una esfera. En el Anexo C se detalla más a fondo el cálculo que permite llegar a esta conclusión.

3.1.2 Esfera de Bloch

El conjunto, o espacios de estados en el que puede "vivir" un *qbit* es comparable a un espacio vectorial complejo bidimensional lo que a priori otorga 4 dimensiones, que quedan reducidas dadas las restricciones

impuestas. Puesto que esto no es muy práctico, comúnmente se establece una biyección entre la superficie de una esfera de radio 1 y el plano complejo $\mathbb{C} \cup \{\infty\}$, es decir, el plano junto con el punto del infinito. Esta variedad se denomina esfera de Bloch (Figura 1) en honor al físico Felix Bloch.

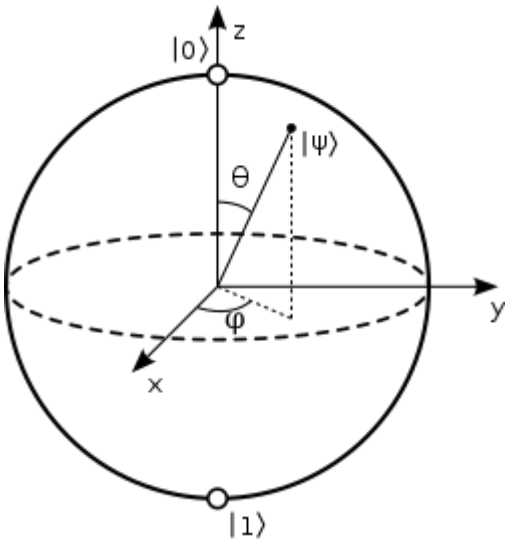

Figura 1: Esfera de Bloch

Dado un estado $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, y puesto que solamente la fase relativa entre los dos coeficientes tiene significado físico (ver Anexo C), se puede tomar el coeficiente de $|0\rangle$ como real y no negativo. Con esta premisa y (1), se puede escribir $|\varphi\rangle$ de esta forma:

$$|\varphi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle = \cos\frac{\theta}{2}|0\rangle + (\cos\phi + i\sin\phi)\sin\frac{\theta}{2}|1\rangle \quad (5)$$

Esto nos deja el *qbit* determinado únicamente con dos parámetros. De esta forma, cada posible estado del *qbit* corresponde a un punto de la superficie de la esfera, lo que le otorga dos grados de libertad locales, latitud y longitud, o en notación de coordenadas esféricas, los ángulos θ y ϕ, con $0 \leq \theta < \pi$ y $0 \leq \phi < 2\pi$.

3.1.3 Casos particulares

También suele denominarse esfera de Bloch al conjunto de estados puros del sistema de un número arbitrario (finito) de niveles. En tal caso, la esfera de Bloch deja de ser una esfera como tal, aunque no por ello pierde su estructura de variedad dimensional, sino que adquiere una estructura conocida como espacio simétrico.

En el caso general de un sistema formado por N *qbits*, el conjunto de estados se determina por un punto en el espacio de Hilbert de dimensión compleja 2^N. Por ejemplo, dado el estado compuesto $|0100\rangle$, se puede obtener como:

$$|0100\rangle = |0\rangle_1 \otimes |1\rangle_2 \otimes |0\rangle_3 \otimes |0\rangle_4$$

Donde cada término hace el papel de espacio vectorial generado por el estado puro y el subíndice entre 1 y 4 representa la posición (el *qbit* en particular).

En el caso de tratarse de sistemas de más de un *qbit*, los estados puros que componen la base del espacio donde viven los elementos son 2^N, uno por cada posible combinación de $|0\rangle$ y $|1\rangle$. Por ejemplo, el estado $|\phi\rangle = \frac{1}{\sqrt{3}}(|001\rangle + |100\rangle + |110\rangle)$ otorga la probabilidad de ⅓ a los estados $|001\rangle$, $|100\rangle$ y $|110\rangle$, y vectorialmente es una combinación lineal de los 8 estados $|000\rangle$, $|001\rangle$, … $|111\rangle$, representado por un vector de 8 coordenadas de coeficientes $\frac{1}{\sqrt{3}}$ en la segunda, tercera y quinta posición.

En la esfera de Bloch vista como espacio de Hilbert, dos puntos a distancia 2 son inmediatamente ortonormales. Por ello, dos puntos diametralmente opuestos cumplen esta condición de distanciamiento y ambos pueden determinar una base del espacio. En particular, los "polos" de la esfera *N* y *S,* en coordenadas cartesianas (0, 0, 1) y (0, 0, -1) respectivamente, equivalen a los autovectores v_1 y v_{-1} correspondientes a los autovalores, positivo y negativo, de la matriz de Pauli σ_z (Anexo D).

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad v_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad v_{-1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

No obstante, todo punto de la esfera se corresponde con un autovector de la proyección del operador determinado por esta matriz de Pauli sobre la dirección que determina dicho punto. En terminología de cuántica, los polos *N* y *S* son básicamente los estados $|0\rangle = 1|0\rangle + 0|1\rangle$ y $|1\rangle = 0|0\rangle + 1|1\rangle$, aunque en terminología de *spin* ½ son comúnmente denominados $|+\rangle$ y $|-\rangle$, o también "*spin* arriba" y "*spin* abajo" (ver Anexo D). Estas propiedades sobre los polos norte y sur también se aplican a los extremos de la esfera situados sobre los otros dos ejes cartesianos y las respectivas matrices de Pauli σ_x y σ_y.

3.2 Trasfondo matemático

Una vez estudiadas las premisas en las que se basa el comportamiento de los *qbits,* podemos transformarlas en elementos y leyes matemáticas más controlables para hacer más sencilla la familiarización con ellos. Esta sección comienza introduciendo conceptos básicos del álgebra, estructuras y espacios vectoriales, para posteriormente poder determinar matemáticamente cómo se comporta un sistema cuántico y qué transformaciones pueden o no realizarse sobre él.

Las temáticas de las secciones siguientes están altamente relacionadas y se complementan progresivamente unas a otras, por lo que su lectura ordenada permitirá al lector comprender todos los conceptos y principios del álgebra vitales para entender la naturaleza de los sistemas cuánticos.

3.2.1 Estructuras algebraicas

Una estructura algebraica es una n-tupla formada por un conjunto no vacío y una serie de operaciones que se aplican sobre elementos del conjunto. Se clasifican según la naturaleza de sus operaciones, que pueden ser leyes de composición interna, externa, o ambas. En esta subsección se recogen las características de los grupos, anillos y cuerpos para poder posteriormente introducir los espacios vectoriales, una estructura más compleja que juega un papel significativo en el estudio de los sistemas cuánticos.

Estas tres estructuras algebraicas comparten algunas características, aunque el concepto de anillo introduce más restricciones al grupo y, a su vez, el concepto de cuerpo es aún más fuerte que el de anillo. La notación y las propiedades se enuncian a continuación:

$$\textbf{Grupo } G = (S, \bullet) \overset{def}{\iff} \begin{cases} 1.\ \forall x, y \in S,\ x \bullet y \in S \\ 2.\ \exists\ e\ \in S :\ \forall x\ \in S, x \bullet e = \ x \bullet e = x \\ 3.\ \forall x\ \in S\ \exists x^{-1} \in S :\ x \bullet x^{-1} = e \\ 4.\ \forall x, y, z\ \in S, (x \bullet y) \bullet z = \ (x \bullet y) \bullet z \end{cases}$$

$$\textbf{Anillo } A = (S, +, \bullet) \overset{def}{\Longleftrightarrow} \begin{cases} 1. \forall x, y \in S, x \bullet y \in S \\ 4. \ \forall x, y, z \in S, (x \bullet y) \bullet z = (x \bullet y) \bullet z \\ 5. (S, +) \ es\ un\ grupo\ abeliano \\ 6. \ \forall x, y, z \in S, \ x \bullet (y + z) = (x \bullet y) + (x \bullet z) \end{cases}$$

$$\textbf{Cuerpo } K = (S, +, \bullet) \overset{def}{\Longleftrightarrow} \begin{cases} 7. (S, +, \bullet) \text{ es un anillo conmutativo unitario} \\ 8. \ \forall x \in S, x \neq 0_S, \ \exists x^{-1} \in S : \ x \bullet x^{-1} = 1_S \end{cases}$$

Donde cada premisa quiere decir:

1. El conjunto S es cerrado bajo la operación "•": los elementos que resultan de aplicar la operación permanecen dentro del conjunto.
2. Existe un elemento neutro (denominado unidad) que no modifica a los demás si se aplica "•" ya sea por la izquierda o por la derecha.
3. Todo elemento tiene un inverso que resulta en la unidad cuando se aplica "•" en cualquier orden.
4. La operación "•" es asociativa.
5. El conjunto (S, +) cumple las condiciones de ser un grupo y, además es conmutativo (el orden de los factores no altera el resultado).
6. La operación "•" es distributiva respecto de "+".
7. Si la operación "•" del anillo (S, +, •) es conmutativa, se denomina anillo conmutativo. El elemento neutro de "+" se denomina "cero" (0_S). Si existe además neutro para "•", éste se denomina unidad (1_S), y el anillo se dice que es unitario.
8. Todo elemento del cuerpo distinto de 0_S tiene inverso multiplicativo.

Cabe destacar que, comúnmente, las operaciones "•" y "+" hacen referencia al producto y suma naturales, pero no siempre ha de ser así, pues los elementos de un grupo pueden no tratarse de números sino de cualquier estructura compleja como vectores, matrices, o funciones. El caso de los grupos de matrices nos interesa especialmente, pues un operador cuántico se puede tratar como una matriz de dimensiones $2^N \times 2^N$, siendo N el número de *qbits*, y el hecho de poder contar con que forman un grupo facilita mucho el estudio de sum comportamiento.

3.2.2 Espacios vectoriales

Un espacio vectorial es una estructura algebraica construida a partir de un conjunto no vacío que presenta una estructura de cuerpo. Los elementos de un espacio vectorial son vectores, y a los elementos del cuerpo se los denomina escalares. Está dotado de una operación interna, denominada "suma", que actúa sobre los elementos del conjunto y otorga como resultado otro elemento (es decir, es cerrada), y una operación externa, llama producto escalar, que se aplica a dos elementos del conjunto para obtener un elemento de cuerpo: un escalar. La fórmula (6) ofrece un ejemplo de un producto escalar clásico entre dos vectores de n coordenadas v y w.

$$\langle v, w \rangle = \sum_{i=1}^{n} v_i \cdot w_i \qquad (6)$$

La operación interna ha de cumplir las propiedades asociativa y conmutativa, y presentar un elemento neutro y existencia de inversos, también denominados opuestos. La operación externa de producto escalar ha de presentar también las propiedades asociativa, conmutativa y distributiva:

$$k\langle v, w \rangle = \langle kv, w \rangle$$

$$\langle v, w \rangle = \langle w, v \rangle$$

$$\langle v, w\rangle + \langle u, w\rangle = \langle v + u, w\rangle$$

Todo espacio vectorial se construye a partir de una base, esto es, el mínimo conjunto de elementos del espacio a partir de los cuales se puede obtener cualquier otro elemento por medio de la suma y el producto por escalares. Es lo que se conoce como combinación lineal, por ejemplo:

$$|\varphi\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle = \alpha\begin{pmatrix}1\\0\end{pmatrix} + \beta\begin{pmatrix}0\\1\end{pmatrix}$$

Siendo en este caso el conjunto ordenado $\{|0\rangle, |1\rangle\}$ la base del espacio, representados respectivamente por los vectores que conforman la matriz unidad. El cardinal de la base determina la dimensión del espacio, lo que también se conoce en geometría como grados de libertad.

Una cualidad del producto escalar es que permite introducir el concepto de ortogonalidad. Dos vectores se dice que son ortogonales respecto de un producto escalar si al aplicarlo sobre ellos el resultado es 0. En la geometría euclídea sobre el plano, dos vectores ortogonales se identifican si son perpendiculares, es decir, si encierran un ángulo de 90 grados. Una base es ortogonal cuando sus elementos son ortogonales entre ellos.

La fórmula dada por (6) representa el producto escalar natural, que se calcula simplemente multiplicando los elementos de los vectores que se encuentren en la misma posición y sumándolos todos. Representado matricialmente, es equivalente al producto del primer vector en forma de fila con el segundo en forma de columna. Pero un producto escalar puede ser un objeto más complejo, pues es en realidad una aplicación bilineal. Si E es un espacio vectorial con coeficientes en un cuerpo K, un producto escalar es una función $f_A: ExE \rightarrow K : f(v, w) = v^T \cdot A \cdot w$, donde v y w son vectores del espacio en columnas y A es una matriz simétrica definida positiva de dimensión n×n, siendo n la dimensión del espacio. La razón por la que la matriz A debe ser simétrica es que, de no ser así, el producto escalar no sería conmutativo. Para el producto escalar definido por (6), la matriz A es la identidad.

El concepto de espacio vectorial puede ser ampliado si se introducen más axiomas y se caracteriza la naturaleza de sus elementos con restricciones más fuertes. Al estar definido un producto escalar, automáticamente puede definirse una norma dada por $\|v\| = \sqrt{\langle v, v\rangle}$. Gracias a que la matriz A es definida positiva, la norma inducida por el producto escalar está bien definida. Cuando dotamos al espacio vectorial de una norma, obtenemos un espacio normado. Una norma permite introducir el concepto de distancia; es un operador que, similarmente al producto escalar, actúa sobre dos vectores y devuelve un número que nos indica como de "lejos" se encuentran los elementos. La distancia entre dos vectores v y w viene dada por:

$$d(v, w) = \|v - w\| = \sqrt{\langle v - w, v - w\rangle} = \sqrt{\|v\|^2 + \|w\|^2 - 2\langle v, w\rangle}$$

Todo espacio normado es además un espacio métrico. La introducción del concepto de norma permite hablar de vectores normales: un vector se dice que es normal si su norma resulta ser la unidad del cuerpo donde viven sus coeficientes. Si una base está formada por vectores ortogonales y de norma 1, se dice que es ortonormal.

Subespacios vectoriales

Un subespacio vectorial es, en el sentido más genérico, un espacio vectorial contenido dentro de otro. Todas las operaciones y leyes que lo rigen son heredadas del espacio contenedor. La dimensión de un subespacio vectorial es estrictamente menor que la del espacio original y, por tanto, su base tiene menos elementos, ya que si tuviera los mismos, estaríamos hablando del mismo espacio. Un ejemplo de subespacio vectorial de $\mathbb{R}^n$ es $\mathbb{R}^k$, con k < n. También son subespacios vectoriales rectas en el plano que pasen por el origen, cuya base la constituirían sus respectivos vectores directores.

Espacios de Hilbert

Los espacios de Hilbert deben su nombre al matemático David Hilbert. Son una generalización del concepto de espacio euclídeo, esto es, un espacio donde se cumplen los axiomas de la geometría de Euclides, aquella que alude a las nociones de geometría que resultan más naturales para el ser humano.

La rama de las matemáticas que estudia los espacios de Hilbert es el análisis funcional. Formalmente hablando, se trata de espacios con un producto interior (producto escalar) bien definido y que son completos con la norma inducida por ese producto interior. Un espacio es completo cuando toda sucesión convergente (de Cauchy) tenga límite y éste se encuentre dentro del espacio.

Estos espacios ayudan a clarificar y generalizar transformaciones lineales como la transformada de Fourier y son de vital importancia en la formulación matemática de las leyes y axiomas de la mecánica cuántica.

3.2.3 Álgebra matricial y aplicaciones

Esta subsección reúne estos elementos para ayudar al lector a comprender algunos aspectos que se han dado por supuestos en anteriores secciones, así como establecer un nexo entre las premisas de la cuántica y su implicación en el desarrollo del algoritmo de Solovay-Kitaev en el sentido empírico.

Una matriz se define por un conjunto de números distribuidos en filas y columnas, llamados entradas (o coeficientes). Comúnmente se denotan por una letra mayúscula $A \in M_{n\times m}(K)$, donde "$n\times m$" denota las dimensiones (filas por columnas), K hace referencia al cuerpo al que pertenecen los coeficientes de la matriz, y $M_{n\times m}(K)$ es la notación general para referirse al conjunto de todas las matrices $n\times m$ con coeficientes en K. Cuando la matriz está formada por una sola columna se denomina vector.

Habitualmente, un conjunto de matrices cumpliendo ciertas propiedades suele formar un anillo, grupo o cuerpo algebraico, dependiendo en cada caso de sus características. Esta subsección introduce los conceptos de matrices unitarias y ortonormales. Estas son de especial interés en los sistemas cuánticos ya que, como veremos a continuación, las puertas cuánticas se pueden representar por matrices unitarias, y cualquier estado posible de un *qbit* o un sistema de *qbits* se identifica con un vector.

Para simplificar las operaciones, los ejemplos que se ilustran tratarán con operaciones sobre un único *qbit*, por lo que los operadores serán matrices 2×2.

Aplicaciones en sistemas cuánticos

En la subsección 3.1.1 se menciona que el modelo de computación que nos interesa estudiar es el del circuito cuántico, en el cual se aplican operadores a los *qbits*. Esto quiere decir que podemos interpretar a un *qbit* como una combinación lineal de dos elementos, los estados puros $|0\rangle$ y $|1\rangle$. Matemáticamente hablando, los estados $|0\rangle$ y $|1\rangle$ se corresponden con los elementos de una base de un espacio vectorial de dos dimensiones, donde los estados compuestos juegan el papel de vectores. Por otro lado, los operadores, o puertas cuánticas, que actúan sobre estos vectores se convierten en matrices, objetos que actúan sobre un *qbit* y devuelven otro. No obstante, debido a que los *qbits* han de responder a ciertas leyes, no cualquier matriz puede considerarse un operador.

Podemos establecer una correspondencia entre todos los posibles estados de un *qbit* $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ y un subespacio vectorial de $\mathbb{C}^2$, donde los estados puros $|0\rangle$ y $|1\rangle$ son los elementos básicos. En notación vectorial estos elementos equivalen a:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}, \qquad |\varphi\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}$$

La razón de que esta base no genere directamente un espacio isomorfo a $\mathbb{C}^2$ es que los coeficientes que acompañan a la base han de cumplir, entre otras restricciones, la ecuación (1). Esto elimina grados de libertad, por lo que su dimensión queda reducida. Como ya se mencionó en la subsección 3.1.2, este espacio es isomorfo a la superficie de una esfera tridimensional. Veamos ahora cómo se define un operador. Sea la matriz 2×2:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} : \; a, b, c, d \in \mathbb{C}$$

Para poder ser considerada un operador, o puerta cuántica, debe poder operar sobre un *qbit* y devolver otro. Si bien una matriz de estas dimensiones aplicada a un vector 2×1 siempre devuelve otro vector 2×1, esto no garantiza que viva en la Esfera de Bloch, pues para que esta comparación entre *qbits* y vectores funcione, no podemos salirnos de las normas de la cuántica. El producto $|\psi\rangle = A|\varphi\rangle$ es otro vector 2×1:

$$|\psi\rangle = \begin{pmatrix} a\alpha + b\beta \\ c\alpha + d\beta \end{pmatrix}$$

Los nuevos coeficientes α' = aα+bβ y β' = cα+dβ son números complejos que determinan la distribución de probabilidad del *qbit* $|\psi\rangle$ y, por tanto, han de cumplir la ecuación (1). Si desarrollamos la ecuación aplicada a estos nuevos coeficientes, el cálculo resulta en un sistema de ecuaciones no lineales con 8 variables complejas que resultan en 16 variables reales, lo cual no es cómodo de manejar para humanos. Pero gracias a la esfera de Bloch, podemos determinar que todo operador ha de mantener el elemento al que aplica dentro de esta esfera, por lo que podemos concluir que una puerta ha de ser una matriz de giro. En coordenadas cartesianas, una matriz de giro de θ radianes alrededor de los ejes x, y, z tiene la siguiente forma:

$$M_{x,\theta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}, \qquad M_{y,\theta} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix},$$

$$M_{z,\theta} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Cualquier combinación de una de ellas para diferentes ángulos también resulta en una matriz de giro, que aplicados a puntos de la superficie de una esfera, los mantiene dentro. El Anexo C ofrece un cambio de variable entre un *qbit* expresado en términos de coeficientes de probabilidad y coordenadas esféricas (coordenadas en $\mathbb{R}^3$ a partir de 2 parámetros). Una vez dadas las coordenadas esféricas, es trivial pasar a coordenadas cartesianas y aplicar las matrices de giro, o puertas cuánticas, a puntos de la esfera de Bloch.

Pero este método añade el coste de un constante cambio de base. Por ello es más interesante conocer cómo se "materializan" estas puertas cuánticas en matrices 2×2 (o $2^N \times 2^N$ para el caso de sistemas de N *qbits*). Es aquí donde entran en juego las matrices unitarias, no sin antes familiarizarnos con el caso particular de las matrices ortonormales.

Matrices ortonormales y unitarias

Una matriz cuadrada, con coeficientes reales, es ortogonal cuando está formada por vectores que forman una base ortogonal, y es ortonormal cuando, además, la norma de todos los vectores es igual a 1.

Otra propiedad interesante de las matrices ortonormales es que el producto de la matriz por su traspuesta da lugar a la matriz identidad, de lo que se deduce que su determinante ha de ser 1 o -1. Por tanto, una matriz es ortonormal si y solo si tiene inversa igual a su traspuesta. Cabe destacar que no es casualidad que las matrices de giro definidas anteriormente cumplan estas propiedades. Pero centrándonos

en matrices reales 2×2, dadas las propiedades descritas de ortonormalidad tenemos las siguientes restricciones:

$$AA^T = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Leftrightarrow \begin{cases} a^2 + b^2 = 1 \\ c^2 + d^2 = 1 \\ ac + bd = 0 \\ ad - cb = \pm 1 \end{cases}$$

Luego cualesquiera coeficientes a, b, c y d que cumplan estas ecuaciones formarán una matriz ortonormal. Pero cuando se trabaja sobre el cuerpo $\mathbb{C}$, no aplican las mismas propiedades de ortonormalidad, sino que en este caso se habla de matrices unitarias. La diferencia respecto de trabajar en $\mathbb{R}$ reside en que la matriz unidad se obtiene de multiplicar la matriz U por su conjugada traspuesta $U^\dagger$, también denominada hermítica, es decir, trasponer la matriz y cambiar de signo la componente imaginaria de sus términos. Luego se tiene:

$$UU^\dagger = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} \bar{a} & \bar{c} \\ \bar{b} & \bar{d} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Leftrightarrow \begin{cases} |a|^2 + |b|^2 = 1 \\ |c|^2 + |d|^2 = 1 \\ \bar{a}c + \bar{b}d = 0 \\ a\bar{c} + b\bar{d} = 0 \end{cases} \quad (7)$$

Al igual que en el caso real, la condición del determinante se puede obtener de las demás ecuaciones. Las primeras ecuaciones vienen de la propiedad de que para todo número complejo z, $z\bar{z} = |z|^2$. La cuarta ecuación es consecuencia de la tercera, pues $a\bar{c} + b\bar{d} = \overline{a\bar{c} + b\bar{d}}$. De la definición se deduce además que una matriz es unitaria si y solo si tiene inversa igual a su traspuesta conjugada. Combinando esto con la regla de la inversa para matrices 2×2 (ver Anexo F), dada una matriz unitaria cualquiera M de determinante ±1:

$$AA^{-1} = AA^\dagger = I \Rightarrow \begin{pmatrix} \pm d & \mp b \\ \mp c & \pm a \end{pmatrix} = \begin{pmatrix} \bar{a} & \bar{c} \\ \bar{b} & \bar{d} \end{pmatrix} \Rightarrow \begin{cases} \bar{a} = \pm d \\ \bar{b} = \mp c \end{cases} \quad (8)$$

Aplicando las reglas del conjugado complejo y multiplicando ambos lados de las ecuaciones de (8) por a y b respectivamente, es fácil comprobar que (7) y (8) son equivalentes.

Si el determinante de una matriz unitaria es 1, se denomina unitaria especial. Para todo entero n, las matrices unitarias de dimensión n con coeficientes en un cuerpo *K* (generalmente $\mathbb{C}$ o $\mathbb{R}$) forman un grupo $U_K(n)$: naturalmente, la identidad es una matriz ortonormal por excelencia, y por definición de matriz unitaria, la inversa de ésta también lo es. Por último, sean dos matrices U, V ∈ $U_K(n)$ y W=UV:

$$(UV)(UV)^\dagger = UVV^\dagger U^\dagger = UIU^\dagger = I \Rightarrow UV = W \in U_K(n)$$

Luego W también es una matriz unitaria, lo que permite concluir que el conjunto es cerrado bajo el producto de matrices, que junto con la existencia de elemento neutro e inverso, que además coincide con el conjugado traspuesto para *K*=$\mathbb{C}$, nos da un grupo.

3.2.4 Grupo SU(2)

La notación $\mathfrak{su}(2)$ es la abreviatura de *Special Unitary* o grupo unitario especial, donde 2 indica la dimensión de la matriz. En otras palabras, se trata de un grupo de matrices unitarias con coeficientes en $\mathbb{C}$ y determinante 1. La subsección anterior permite concluir que estas matrices tienen la particularidad de que equivalen a puertas lógicas en computación cuántica. Cabe destacar que se trata de un grupo infinito, pues existe un número infinito de combinaciones de elementos en $\mathbb{C}$ que pueden dar lugar a una matriz de

estas características. La dimensión de $\mathfrak{su}(n)$ es n^2-1, por lo que $\mathfrak{su}(2)$ es un espacio de dimensión 3 que, además es homeomorfo a una 3-esfera.

Este grupo es de especial interés ya que es el componente básico del funcionamiento del algoritmo de Solovay-Kitaev y el teorema con el mismo nombre. Conocer su estructura y propiedades es una parte vital del proceso de análisis para, posteriormente, poder implementar el algoritmo y sacar conclusiones. Uno de los factores que afecta a la implementación es hacer una buena elección de un subgrupo (finito) de $\mathfrak{su}(2)$ ya que, naturalmente, trabajar computacionalmente con un conjunto infinito no es viable.

Antes de enunciar el teorema de Solovay-Kitaev, es conveniente familiarizarse con subgrupos y con el concepto de densidad. Sea G = (S, ·) un grupo. Sea R⊆S. Se dice que H = (R, ·) es un subgrupo de G (H⊆G) si:

- R es cerrado bajo la operación "·"
- Todo elemento de R tiene un inverso también dentro de R
- La unidad del grupo G está en R

Si además R⊂S (R está estrictamente contenido en S), se denomina subgrupo propio.

Un subgrupo H⊆G se dice que es normal si el conjunto de las clases por la izquierda {x·G : x∈H} coincide con el conjunto de clases por la derecha {G·x : x∈H}. De esta definición se deduce que todo subgrupo normal es abeliano.

Al tratarse de un conjunto de matrices, $\mathfrak{su}(2)$ tiene la particularidad de que podemos dotarlo de una norma y trabajar así sobre un espacio topológico y, además, presenta otras características que permiten que se trate de una variedad diferenciable de dimensión 3, un concepto más fuerte que el de espacio métrico o topológico (toda variedad diferenciable es un espacio topológico, y todo espacio topológico es métrico). En la subsección 4.2.3 se detallan los cálculos de normas para matrices.

Ahora podemos introducir el concepto de densidad: dado un espacio topológico (X, T), un conjunto A⊆X es denso si la clausura (o cierre) de A coincide con el conjunto X. La clausura de un conjunto se define como el conjunto cerrado más pequeño, en el sentido de la inclusión, que lo contiene. En otras palabras, el conjunto A es denso si puede otorgar, para todo miembro de X, un elemento tan próximo como se desee. La elección y obtención de un subgrupo denso de $\mathfrak{su}(2)$ es el factor clave que determina la eficiencia y la eficacia del algoritmo.

3.2.5 Subgrupo conmutador

Dado un grupo (G, ·), el subgrupo conmutador de G es el generado por elementos de la forma $[a, b] = a \cdot b \cdot a^{-1} \cdot b^{-1}$, para todo par de elementos a, b ∈ G. También se denomina subgrupo derivado y se denota por G' o [G, G]. Tiene las siguientes propiedades:

- G' es el grupo trivial {e} si y sólo si G es abeliano (conmutativo)
- El inverso del conmutador $[a, b] = [a^{-1}, b^{-1}]$ también es un conmutador y coinciden si G es abeliano
- G' es un subgrupo normal en G, y el cociente $G/_{[G,G]}$ es abeliano, siendo [G, G] el menor subgrupo que verifica esta propiedad.

Según se postula en [14], aplicado a grupos de matrices, la descomposición de una matriz como grupo conmutador permite, dada una matriz Δ talque $d(\Delta, I) < \varepsilon$, obtener matrices V y W talque $V \cdot W \cdot V^{\dagger} \cdot W^{\dagger} = \Delta$. Si además se da que $d(W, I)$, $d(V, I) < c\sqrt{\varepsilon}$ para alguna constante c, a esta descomposición se le denomina grupo conmutador balanceado. En la sección siguiente se concreta más a

fondo las implicaciones de estas propiedades a la hora de implementar al algoritmo y cómo éstas permiten, en cada paso, obtener mejores aproximaciones a la puerta deseada.

3.3 Teorema y algoritmo de Solovay-Kitaev

Desde un punto de vista matemático, el teorema de Solovay-Kitaev es una declaración sobre la rapidez con la que el grupo $\mathbb{SU}(2)$ puede ser cubierto a partir de un conjunto de puertas dado. Se trata de uno de los resultados más importantes en el campo de la computación cuántica. Desde un punto de vista práctico, es importante tener en cuenta tanto la eficacia como la eficiencia a la hora de diseñar un método que obtenga puertas a priori arbitrarias para poder asegurar una buena tolerancia a errores. Robert M. Solovay y Alexei Kitaev demostraron conjuntamente en su publicación [14] que, si un conjunto de puertas cuánticas genera un subgrupo denso de $\mathbb{SU}(2)$, entonces el grupo puede ser cubierto rápidamente, lo que garantiza una buena aproximación a cualquier puerta deseada usando secuencias suficientemente cortas de puertas del subgrupo generado.

Según [14], el enunciado del teorema para sistemas de un único *qbit* es el que sigue: sea G un subgrupo de $\mathbb{SU}(2)$, y sea $\varepsilon > 0$. Entonces existe una constante κ tal que para toda puerta $U \in \mathbb{SU}(2)$ existe una secuencia S finita de puertas de G de longitud $O(log^{\kappa}(1/\varepsilon))$ tal que $d(U,S) < \varepsilon$.

Diferentes pruebas han dado lugar a diferentes valores de κ, que ronda el valor de 3.97. Un asunto del que el teorema no se preocupa es la eficiencia con la que esta secuencia es hallada en un computador clásico. El algoritmo Solovay-Kitaev ofrece una demostración del teorema y muestra además que dicha secuencia S puede ser obtenida en un tiempo de $O(log^{2.71}(1/\varepsilon))$. Aunque a priori puede resultar contradictorio que el tiempo de ejecución sea menor asintóticamente que la longitud de la secuencia, en el siguiente capítulo se muestra que existen formas de reducir el tiempo gracias a la eliminación de redundancias durante la generación.

3.3.1 Pseudocódigo y recursión

El algoritmo Solovay-Kitaev (o simplemente algoritmo SK) para sistemas de un único *qbit* puede ser expresado en pocas líneas de pseudocódigo:

```
1    Operator solovayKitaev(Operator U, depth n) {
2
3         if (n == 0) {
4             return bassicApproximation(U)
5         }
6         U_{n-1} = solovayKitaev(U, n-1)
7
8         V, W = GCDecompose(U·U_{n-1}†)
9
10        V_{n-1} = solovayKitaev(V, n-1)
11        W_{n-1} = solovayKitaev(W, n-1)
12
13        U_n = V_{n-1}·W_{n-1}·V_{n-1}†·W_{n-1}†·U_{n-1}
14        return U_n
15
16   }
```

Ahora veremos en detalle lo que representa cada una de estas líneas. En primer lugar la declaración `function solovayKitaev(Operator U, depth n)` indica que se trata de una función que recibe dos parámetros de entrada y devuelve como salida un operador. El parámetro `Operator U` hace referencia a la puerta que se desea aproximar, y `depth n` a la profundidad máxima de recursión a la que queremos que itere el algoritmo.

El algoritmo ha de devolver una secuencia de puertas que correspondan a una ε_n-aproximación a la puerta U, siendo ε_n una tolerancia dada perteneciente a una cadena decreciente en n, de tal manera que $\lim_{n\to\infty} \varepsilon_n = 0$.

Las líneas de la 3 a la 5 constituyen el caso base de la recursión, la cual acaba cuando n llega a cero. Naturalmente el algoritmo espera que el valor de n recibido sea un numero entero mayor o igual que cero, pues en otro caso jamás acabaría. La sentencia `bassicApproximation(U)` recibe como entrada la puerta original y devuelve el elemento más próximo del subgrupo denso de $\mathbb{SU}(2)$ previamente generado y cargado en memoria.

En niveles de recursión más profundos, para encontrar una ε_n-aproximación a la puerta `U`, se busca una ε_{n-1}-aproximación `U_{n-1} = solovayKitaev(U, n-1)` haciendo que el algoritmo se llame a sí mismo un total de n veces hasta llegar al caso base. En este paso es necesario asumir que el subgrupo de partida es capaz de encontrar una aproximación suficientemente buena para cualquier puerta de entrada. La puerta U_{n-1} servirá para aproximar la puerta `U` en pasos posteriores. Llamaremos Δ al operador $UU_{n-1}^{\dagger}$. Cabe destacar que, al ser U_{n-1} una ε_{n-1}-aproximación de `U`, Δ es una ε_{n-1}-aproximación de la identidad. Por tanto, los siguientes pasos consisten en aproximar la identidad con una tolerancia de ε_{n-1} y así combinarla con U_{n-1} y obtener una ε_n-aproximación a `U`.

El siguiente paso (línea 8) consiste en descomponer Δ como un grupo conmutador balanceado de operadores unitarios V y W de tal manera que $\Delta = V\cdot W\cdot V^{\dagger}\cdot W^{\dagger}$. Debemos asumir en este punto que dadas las propiedades comentadas en la subsección 3.2.3, si se parte de que $d(\Delta, I) < \varepsilon_{n-1}$, esta descomposición existe, y además existe una constante c tal que $d(W, I), d(V, I) < c\sqrt{\varepsilon_{n-1}}$. En [14] se han dado valores para $c \approx 1/\sqrt{2}$.

Las líneas 10 y 11 se centran en llevar a cabo tareas similares a la línea 6: buscar respectivamente ε_{n-1}-aproximaciones a `V` y `W` llamando recursivamente al algoritmo. Existen pruebas en [14] que afirman que la descomposición en el grupo conmutador balanceado resulta ser una $\varepsilon_n = c'\sqrt{\varepsilon_{n-1}}^3$-aproximación a Δ, para alguna constante c'. Sustituyendo, es fácil ver que, si $\varepsilon_{n-1} < 1/c'^2$, se tiene que $\varepsilon_n < \varepsilon_{n-1}$, confirmando que se trata de una sucesión decreciente. La constante c' es determinante a la hora de escoger el valor de ε_0 necesario para la aproximación del caso base. En particular, en esta construcción, para poder garantizar una sucesión decreciente, debemos tener $\varepsilon_0 < 1/c'^2$. Se han logrado establecer valores de $c' \approx 8c \approx 4\sqrt{2}$, lo que da una cota superior para ε_0 de $1/32$.

Finalmente, el algoritmo concluye en las líneas 13 y 14, multiplicando la cadena obtenida de la ε_{n-1}-aproximación por grupo conmutador de $\Delta = UU_{n-1}^{\dagger}$ con la ε_{n-1}-aproximación inicial de U_{n-1} y devolviendo este valor como ε_n-aproximación de `U`, donde $\varepsilon_n = c'\sqrt{\varepsilon_{n-1}}^3 = 4\sqrt{2}\sqrt{\varepsilon_{n-1}}^3$. Gracias a los valores de estas constantes y a las líneas de pseudocódigo podemos plantear las siguientes recursiones:

$$\varepsilon_n = c'\sqrt{\varepsilon_{n-1}}^3 \implies \varepsilon_n = \frac{1}{c'^2}\left(\varepsilon_0 \cdot c'^2\right)^{3/2^n} \tag{9}$$

$$l_n = 5l_{n-1} \implies l_n = O(5^n) \tag{10}$$

$$t_n \leq 3t_{n-1} + cte \implies t_n = O(3^n) \tag{11}$$

Donde ε_n, l_n y t_n son, respectivamente, la tolerancia, longitud de cadena, y tiempo de ejecución del resultado obtenido en la iteración `solovayKitaev(…, depth n)`. Si despejamos en (9) el valor de n en función de ε_n y ε_0, se llega a la conclusión de que para obtener una tolerancia ε, son necesarias un número de iteraciones

$$n = \left\lceil \frac{\ln\left(\frac{\ln\left(1/\varepsilon \cdot c^2\right)}{\ln\left(1/\varepsilon_0 \cdot c^2\right)}\right)}{\ln\left(3/2\right)} \right\rceil = \left\lceil \frac{\ln\left(\frac{\ln(\varepsilon \cdot c^2)}{\ln(\varepsilon_0 \cdot c^2)}\right)}{\ln\left(3/2\right)} \right\rceil \tag{12}$$

Combinando (12) con (10) y (11) obtenemos una expresión para el comportamiento asintótico del tiempo y la longitud de la secuencia en función de ε en lugar de n.

$$l_n = O(5^n) = O\left(ln^{\ln 5/\ln(3/2)}\left(1/\varepsilon\right)\right) = O\left(ln^{3.97}\left(1/\varepsilon\right)\right) \tag{13}$$

$$t_n = O(3^n) = O\left(ln^{\ln 3/\ln(3/2)}\left(1/\varepsilon\right)\right) = O\left(ln^{2.71}\left(1/\varepsilon\right)\right) \tag{14}$$

Para concluir el análisis, en [14] se advierte del hecho de que no se ha tenido en cuenta la precisión con la que las operaciones aritméticas se llevan a cabo, es decir, en la práctica, ciertas operaciones unitarias no pueden ser calculadas con absoluta certeza. Como consecuencia, se debe trabajar siempre con aproximaciones a tales resultados de dichas operaciones y realizar los cálculos con aritmética de precisión finita. Aunque un minucioso análisis del algoritmo SK habría de requerir tener en cuenta el coste extra de estas inexactitudes, no existe necesidad de realizarlo por el momento, pues se ha logrado comprobar que para los valores de precisión con los que se trabaja en la práctica, la aritmética de coma flotante produce resultados suficientemente satisfactorios.

4 Implementación del algoritmo

Una vez analizado y comprendido el funcionamiento del algoritmo, comenzó la fase de diseño. El primer paso de esta etapa consistió en investigar la existencia de alguna librería o base que sirviera como soporte y, junto a diversos artículos se encontró el repositorio público *skc-python* ([15]). Toda la implementación realizada parte de este código como base.

4.1 Librería original

Este repositorio, implementado por el Dr. Paul Pham, está diseñado como un conjunto de módulos de *Python*, mezclado con algunos *scripts* ejecutables dependientes unos de otros en una jerarquía de capas. La última contribución al repositorio incluía un *script* principal incompleto que ejecutaba el algoritmo para una matriz previamente establecida. Pero este proceso requería de unas operaciones previas para poder trabajar, entre ellas generar un subgrupo denso de $\mathbb{SU}(2)$ como conjunto de partida.

4.1.1 Generación de SU2

Como en cualquier grupo algebraico, todo subgrupo necesita un conjunto original de generadores, al que llamaremos *i-set*. A partir de los operadores de puertas T, H (Hadamard) y T inversa (fórmula (15)), previamente configurados, el *script* generaba todas las demás puertas del subgrupo combinando estas tres hasta una longitud máxima dada. En cada nivel se añade un término más a la cadena, hasta un máximo de 16. Estas cadenas, que resultan de multiplicar las matrices del *i-set*, son las que adoptarán el rol de aproximaciones básicas.

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}, \qquad H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad T^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\frac{\pi}{4}} \end{pmatrix} \tag{15}$$

Para poder generar una cadena de longitud L, cada nivel hace uso de las cadenas de longitud L-1 generadas en el nivel anterior, añadiendo a cada una de ellas cada uno de los 3 elementos del *i-set*. De esta forma el número de cadenas nuevas se triplica en cada nivel. Una vez obtenidas todas las aproximaciones básicas, éstas se agrupaban en un mismo fichero para obtener el subgrupo completo.

Durante el proceso, algunos operadores podían estar aparentemente repetidos, es decir, expresados de más de una forma como combinación de elementos del *i-set*, por ejemplo, la puerta T^8 resulta en la identidad. Estas redundancias entorpecen y ralentizan el posterior proceso de búsqueda. Para evitar esto, antes de añadir un elemento al grupo final se realiza una comprobación y, si se encontraba demasiado próximo a algún operador ya presente, se descartaba. Debido al coste extra que este proceso suponía (una búsqueda lineal extra en cada paso), se optó por omitirlo y buscar otros métodos de eliminación de redundancias que se comentarán en el capítulo de resultados.

No obstante, durante la generación de aproximaciones básicas, un proceso de simplificación se encarga de prevenir elementos duplicados para conjuntos de cadenas de la misma longitud. Esto se consigue gracias a reglas de simplificación establecidas previamente antes de la generación, al igual que el *i-set* y otros parámetros de configuración de $\mathbb{SU}(2)$.

4.1.2 Reglas de simplificación

Dado que las aproximaciones se generan a partir de la combinación de elementos de un set anterior con cada uno de los elementos del *i-set*, se espera que haya un total de 3^L elementos de longitud L. Al ser

16 la longitud máxima generada, este último conjunto estaría formado por 3^{16} = 43046721 elementos. Al juntarlos todos para conformar el subgrupo, esto nos daría un total de

$$\sum_{l=1}^{16} 3^l = \frac{3^{17} - 3}{3 - 1} = 64570080$$

lo cual es una cantidad de elementos inmanejable para mantener en memoria. Pero gracias a las reglas de simplificación, en cada nivel se eliminan elementos duplicados, en el sentido literal, pues dentro de un grupo no existe una única manera de obtener un elemento a partir de la operación básica. Esto permite obtener una cantidad mucho más manejable de elementos, exactamente 256214 para L=16, del orden de 100 veces menos que la cantidad teórica. La Figura 2 muestra, para longitudes de la cadena (operador) desde 1 a 16, el tamaño esperado del grupo y el obtenido gracias a estas reglas.

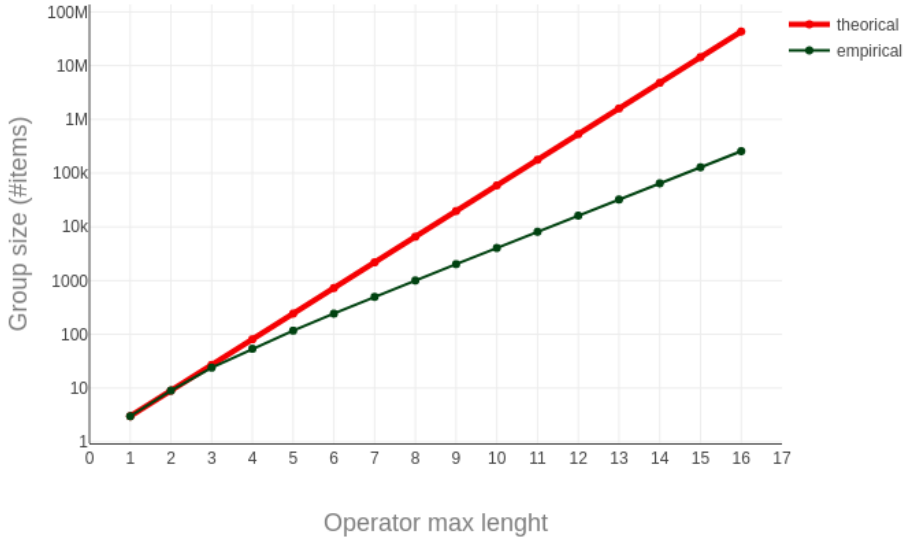

Figura 2: Tamaño teórico y empírico de los subgrupos

4.1.3 Generación del árbol-KD

En ciencias de la computación, un árbol-KD (abreviatura de árbol k-dimensional) es una estructura de datos de particionado del espacio que organiza los puntos en un espacio euclídeo de k dimensiones.

Los árboles-KD son un caso especial de los árboles BSP (Binary Space Partitioning o Partición Binaria del Espacio). Su particularidad reside en que emplean solo planos perpendiculares a uno de los ejes del sistema de coordenadas, mientras que en los árboles BSP los planos pueden ser arbitrarios. Además, los árboles-KD almacenan puntos en todos los nodos y no únicamente en las hojas y, por este motivo, todos los planos han de pasar a través de alguno de los elementos.

Dado un conjunto de n puntos de un espacio euclídeo, la construcción de un árbol-KD conlleva un coste de $O(n\ log(n))$, pero permite encontrar vecinos próximos con una complejidad de $O(n\ log(n))$ en el peor caso, mucho mejor que la búsqueda lineal $O(n)$.

El módulo encargado de la generación del árbol forma parte de la librería original. Recibe como fuente todos los elementos del subgrupo en una lista y permanece en memoria para las operaciones de búsqueda de vecinos próximos. Debido al coste de su construcción, para la ejecución del algoritmo se parte de un árbol previamente serializado en disco de la misma forma que se hacía con el grupo en bruto.

Tanto el árbol como el grupo en forma de lista deben estar presentes en memoria en tiempo de ejecución para que el algoritmo encuentre el elemento más cercano en el caso base de la recursión. Para no tener que implementar una versión del algoritmo por cada método de búsqueda, se implementaron clases adicionales para sacar partido del polimorfismo y encapsular las operaciones a más bajo nivel.

4.2 Clases principales de extensión

Aunque la librería original contaba con documentación, la tarea de ejecutar el algoritmo resultaba ser confusa al principio. Fue necesario un estudio del código y de cómo estaba organizado para garantizar su correcto funcionamiento y posibles puntos de fallo. Cada módulo tenía una función concreta (generar subgrupos, generar un árbol de búsqueda, etc.) o bien contenía ficheros con métodos útiles (operaciones entre matrices, cálculo de distancias, etc.). Muchos de los comportamientos eran desconocidos e inexplicables debido al alto acoplamiento entre módulos y poco uso de orientación a objetos, ya que la mayor parte del código era funcional.

Las partes más relacionadas directamente con el algoritmo son los métodos de búsqueda del vecino más próximo para el caso base, el cálculo de la distancia entre dos matrices y la descomposición en el grupo conmutador. La librería contaba con más de una forma de llevar a cabo estas tareas. Las siguientes secciones describen sus diferencias y cómo el polimorfismo permite que el algoritmo pueda trabajar con cualquier combinación de estos métodos sin afectar a la implementación.

4.2.1 Approxes Finder

Este módulo se engloba las clases encargadas de buscar la mejor aproximación para el caso base del algoritmo, donde finaliza la recursión. Los métodos posibles son búsqueda lineal o por árbol-KD. Cada método de búsqueda es encapsulado por una clase que extiende de *OperatorApproxesFinder*. Ambas clases implementan dos métodos principales, uno para inicializar el subgrupo y otro para encontrar la mejor aproximación de un elemento dado.

La clase *BasicApproxesFinder* recibe por constructor el fichero que contiene el subgrupo de partida serializado. En el método de inicialización carga el subgrupo en memoria en forma de lista para poder encontrar la mejor aproximación bajo demanda en llamadas posteriores al método principal. Existe también una clase que extiende de esta para buscar linealmente un elemento en un grupo generado aleatoriamente. Su utilidad será comentada en capítulos posteriores.

KDTreeApproxesFinder es la otra clase que extiende a *OperatorApproxesFinder*. También recibe de entrada el fichero con los datos serializados, en este caso, un árbol-KD. La inicialización consiste en cargar el objeto árbol en memoria para, posteriormente, llamarlo para buscar la mejor aproximación.

4.2.2 Factor Method

Aquí residen las clases que se encargan de la descomposición de un operador en su grupo conmutador. Las clases de este módulo extienden de *OperatorFactorMethod,* cuyo método principal recibe de entrada un operador y, haciendo uso de la librería original devuelve los componentes de su grupo conmutador según el método de Dawson u otro llamado *aram_diagonal_factor*.

En las primeras ejecuciones del código original se comprobó que el método de Dawson es mucho más efectivo y preciso. Dado que la forma de descomponer un operador no entra dentro de las competencias del estudio del algoritmo, todas las pruebas fueron realizadas utilizando el método de Dawson.

4.2.3 Distance

La manera de medir la distancia entre dos operadores puede determinar de forma muy directa la eficacia del algoritmo. A diferencia de los demás, al no requerir ningún tipo de configuración inicial, este módulo está conformado por clases estáticas. El método encargado de medir la distancia entre dos matrices sólo recibe a éstas como entrada y devuelve un escalar.

La clase *TraceDistance* realiza el cálculo de la distancia según una variante de la norma euclídea, denominada espectral cuando se trata de matrices. Dadas dos matrices A, B ∈ $M_{n\times n}(\mathbb{C})$, se calcula de la siguiente manera:

$$d(A,B) = \left\|\vec{\lambda}\right\|_2 : \lambda_i \in \text{autovalores}\left((A-B)(A-B)^\dagger\right)$$

mientras que la norma espectral solo hace uso del máximo de los autovalores de la matriz semidefinida positiva $(A-B)(A-B)^\dagger$ y halla su raíz cuadrada.

Partiendo de las mismas premisas, la distancia de Fowler utilizada en la clase *FowlerDistance* calcula:

$$d(A,B) = \sqrt[2]{\left|\frac{n - \left\|tr(A^\dagger B)\right\|}{n}\right|} \leq 1 \qquad (16)$$

Es inmediato comprobar en (16) que esta distancia está acotada entre 0 y 1. Dado que, al igual que para el caso de la descomposición en el grupo conmutador, la norma de matrices no es parte de las competencias del algoritmo, todas las pruebas fueron realizadas utilizando la distancia de Fowler. Además, al tratarse de distancias acotadas, dado que estamos en un espacio de dimensión finita, ambas son equivalentes.

4.3 Módulo principal

Las secciones anteriores han descrito la implementación de las herramientas necesarias que el algoritmo necesita para realizar sus cálculos y operaciones. Una vez probadas e implementadas, se crearon módulos que se centran únicamente en configurar y combinar estas herramientas para poder recoger los resultados deseados. Se hará distinción entre la mera ejecución y recogida de resultados y el módulo encargado de interpretarlos y convertirlos en gráficas para analizar y comparar resultados.

4.3.1 Versiones del algoritmo

Según se describe en [14], el algoritmo hace iteraciones hasta obtener una tolerancia deseada. También se describe el proceso para poder obtener el número de iteraciones necesarias para obtener dicha tolerancia, así como la máxima tolerancia que podemos obtener dado un valor n de iteraciones.

Gracias a esta flexibilidad, fueron implementadas dos versiones del algoritmo: una que opera dado el nivel máximo de recursión y otra que itera hasta conseguir la tolerancia deseada. No obstante, ambos parámetros están estrictamente ligados, es decir, de acuerdo la ecuación (12), podemos determinar el valor de n a partir de ε_0, y al contrario según (9). De esta forma, conocidos los valores de c y de ε_0, podemos obtener el número de iteraciones necesarias para obtener una tolerancia deseada, y la tolerancia máxima a la que podemos aspirar según las iteraciones que se realicen.

La segunda versión del algoritmo requiere poder calcular la precisión del operador obtenido en cada paso para determinar cuándo se detiene. Si tratásemos la cualidad recursiva del algoritmo como un árbol, en cada paso se generan tres ramas nuevas, hasta que se alcanza el caso base en una de ellas y entonces "retrocede", similar al proceso de búsqueda en profundidad. Por esta razón es imposible determinar en cada paso la precisión que se va a tener sin hacer uso de recursos como variables globales o procesos multi

hilo así que, en cada paso, el algoritmo sencillamente calcula la tolerancia del siguiente nivel a partir de la parte izquierda de la ecuación (9), y entra en el caso base cuando la tolerancia resulta ser menor que ε_0.

Esta versión se implementó con fines más didácticos que experimentales, por ello no tomó parte en la ejecución y recogida de resultados.

4.3.2 Módulo de ejecución

Para la ejecución y recogida de pruebas se implementó una clase encargada de realizar diversas llamadas al algoritmo con parámetros configurables (*SolovayKitaevExecutor*) y un *script* con tres métodos básicos que hacen llamadas a esta clase. Éstos últimos también se encargan de que en cada prueba sea ejecutada para las mismas matrices, generadas aleatoriamente sólo la primera vez y reutilizadas para asegurar la veracidad de los resultados, así como de establecer el directorio de los ficheros de salida.

Para no saturar el *buffer*, los ficheros van siendo sobrescritos en cada ciclo. La clase *SolovayKitaevExecutor* se encarga de las siguientes tareas: iterar las veces indicadas con matrices diferentes para después obtener una media, llamar al algoritmo con los distintos niveles de profundidad indicados y volcar en fichero, para cada nivel, el tiempo y el error medio. El *script* sencillamente llama a esta clase variando el número de matrices diferentes, el nivel de recursión máximo y el método para el caso base, esto es, un objeto *ApproxesFinder* polimórfico.

4.3.3 Módulo de generación de gráficos

Esta parte de la implementación es la más extensa de entre los tres módulos que extienden la librería principal. Los resultados volcados en fichero están recogidos de manera muy uniforme y ordenada para su fácil identificación. No obstante, con los datos de un único fichero no basta para generar y visualizar resultados interesantes, sino que a veces es necesario contar con ejes adicionales, es decir, combinar resultados de varios ficheros para representar en uno de los ejes, por ejemplo, la densidad del grupo, o aunar en una misma gráfica resultados con dos métodos de búsqueda de aproximaciones diferentes.

Para cada uno de los casos particulares se implementó un método que recibía como parámetro el directorio de uno o varios ficheros y los traducía de un modo u otro dependiendo de lo que se quisiera representar en cada caso, ya fuese o bien agrupando por nivel de recursión, o por tamaño del grupo, etc.

Del mismo modo, para cada caso particular de gráfica que se quisiera representar se implementó un *script* diferente que se encargaba de traducir el/los fichero/s necesarios e iterar sobre los datos para transformarlos en objetos interpretables por la librería generadora de gráficas *Plotly* ([16]). Los objetos que proporciona su interfaz y todas las llamadas directas a esta librería fueron encapsulados en otro módulo para que los *scripts* tuviesen que llamar únicamente a éste y no tener que instanciar directamente objetos de *Plotly*.

5 Resultados y validación del teorema

Una vez finalizada la implementación de todos los módulos necesarios se procedió a la ejecución del algoritmo llevando a cabo diversas mediciones. Este capítulo recoge en forma de gráficas un estudio comparativo de la eficiencia y eficacia que presenta el algoritmo a medida que se combinan y varían algunas de sus características y/o propiedades tanto intrínsecas como extrínsecas.

5.1 Gráficas

Para la representación gráfica se hizo uso de una librería de *Python* llamada *Plotly*, que permite configurar el formato y las características necesarias a partir de una sintaxis muy simple en base a tipos básicos de *Python* (listas y diccionarios). Gracias a la versión *offline* las imágenes podían ser generadas como ficheros en local y no fue necesario migrar todo el proyecto a una *shell* interactiva.

A lo largo de esta sección se expondrán los gráficos de resultados agrupados según se centren en analizar eficacia o eficiencia, o según otras componentes variables.

5.1.1 Perspectiva general

Todas las pruebas realizadas se han hecho sobre la primera versión del algoritmo, es decir, la que recibe un nivel máximo de profundidad en la recursión y busca la mejor aproximación, usando en todos los casos la distancia de Fowler y el método Dawson para la factorización en el grupo conmutador. La razón de esta decisión es, como ya se comentó en la sección 4.2, que ambas distancias resultan ser equivalentes y el método *aram_diagonal_factor* no resultó ser muy eficiente en sus cálculos, mientras que el método de búsqueda de aproximaciones sí resulta determinante en la eficiencia.

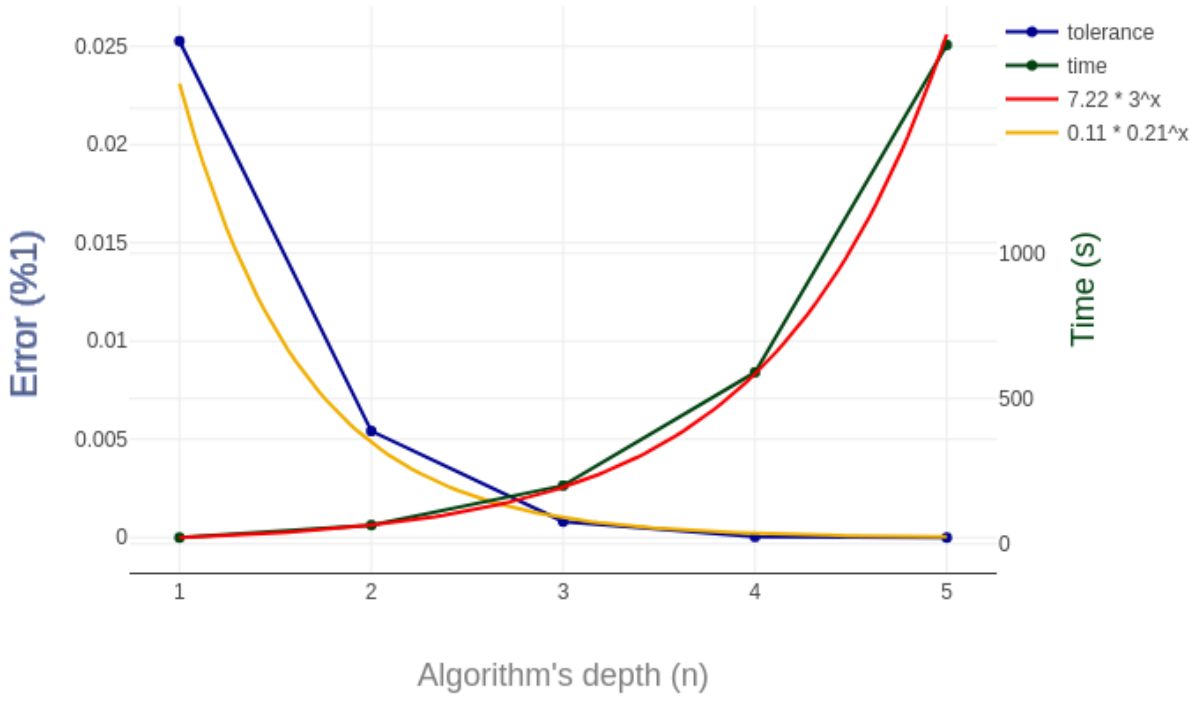

Figura 3: Resultados para el buscador básico con el mayor subgrupo

En primer lugar, se muestra un ejemplo básico (ver Figura 3). Servirá para familiarizar al lector con las unidades utilizadas y la forma de presentar los datos, que se mantendrá en el resto de figuras. En este caso el eje de abscisas representa la profundidad del algoritmo (el máximo nivel de recursión). Aparecen dos ejes de ordenadas, uno para el tiempo (eficiencia), en verde, y otro para la tolerancia, o error obtenido (eficacia), en azul. La prueba se llevó a cabo usando búsqueda lineal sobre el subgrupo más denso generado, aproximando un total de 2 matrices para luego obtener una media. Como es de esperar, a medida que se exige más nivel de recursión, el tiempo aumenta y el error obtenido disminuye, ambos exponencialmente. Para comprobar el crecimiento asintótico se ha combinado la gráfica con funciones exponenciales calculados obteniendo la curva exponencial de regresión pertinente a partir de los datos.

A lo largo de las siguientes secciones, para facilitar la lectura de las gráficas asumiremos que se trata de un crecimiento exponencial, por lo que no será necesario combinar los resultados con tales funciones.

5.1.2 Comparación entre métodos de búsqueda

Como ya se comentó en la subsección 3.3.1, el caso base del algoritmo consiste en buscar una aproximación básica dentro del subgrupo para el elemento dado. Esta parte es la que alberga mayor coste computacional si ignoramos las operaciones de multiplicación al finalizar cada nivel.

Para ello existen dos estrategias: búsqueda lineal y con árbol-KD. Sobre las mimas matrices se probó a ejecutar el algoritmo con ambos métodos (Figura 4). Como antes, tenemos por un lado el tiempo (en verde) y el error (en azul), tanto para la búsqueda lineal (en oscuro) y con árbol (en claro).

Debido al excesivo tiempo de la búsqueda lineal, para la búsqueda en árbol se llegó hasta un nivel de recursión de 5 y se hizo la media de tres matrices, mientras que para la búsqueda lineal fue solo hasta nivel 4 y únicamente dos de las tres matrices participaron en el promedio. Fue necesario usar estos resultados concretos porque se corresponden a los mismos operadores (aleatorios para cada ejecución).

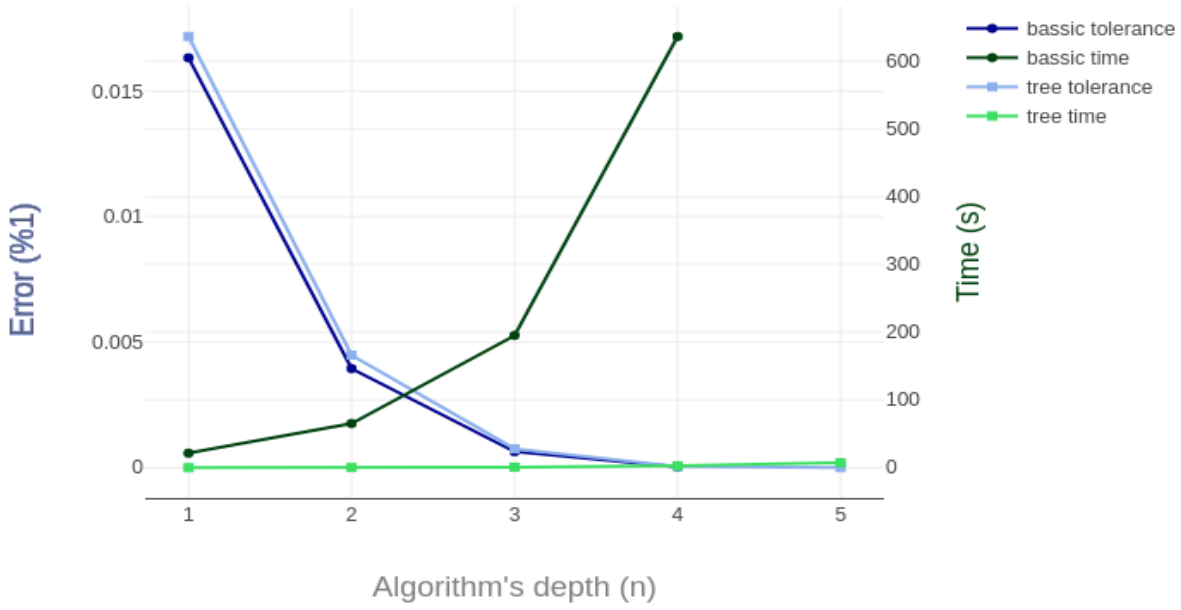

Figura 4: Resultados para búsqueda lineal vs. árbol-KD para el mayor subgrupo

Mientras que, dado que se utilizaron las mismas matrices de entrada, el error es exactamente el mismo (ignorando que uno fue tomado de una media de tres elementos y otro de sólo dos de ellos, pues en el fichero de salida se aprecia que son exactos), el tiempo de ejecución presenta una diferencia asintótica importante. Este gráfico comprueba que el uso de un árbol de búsqueda produce los mismos resultados en un tiempo minúsculo y con crecimiento lineal con una pendiente casi despreciable a estos niveles de recursión. Nótese que para n=4, el error es del orden de 10^{-6}. En gráficas posteriores se podrá apreciar este valor con más detalle.

5.1.3 Cambios en la densidad del grupo

Una vez comprobado el alto coste de la búsqueda lineal, surgió plantearse formas de hacer este proceso más corto sacrificando la menor eficacia posible. Una de las posibles maneras de lograr este objetivo es reducir el tamaño del conjunto de elementos básicos, es decir, reducir el subgrupo.

Esto plantea una nueva incógnita que es qué criterio es el más correcto para decidir qué elementos forman parte del grupo y cuáles pueden ser ignorados al ser 'redundantes'. Para poder asegurar una distribución uniforme y conservar la densidad (topológicamente hablando), lo más correcto es considerar que un elemento está "repetido" si se encuentra lo suficientemente cerca de algún otro que ya sea miembro.

Si bien este proceso puede parecer el óptimo, añade un coste computacional de $O(n^2)$ a la hora de generar el subgrupo reducido, ya sea calculando las distancias una por una por parejas (un total de $\frac{n(n-1)}{2}$ operaciones básicas), o calculando los k vecinos próximos de los miembros y descartando al elemento candidato cuando se encuentre entre éstos.

No obstante, existe un término medio que consiste en limitar la longitud de las cadenas a la hora de generar el grupo a partir del *i-set*. Al existir 3 elementos en éste, para cada nivel de longitud se espera que el grupo tenga el triple de elementos que el anterior, pero, como ilustra la subsección 4.1.2, gracias a las reglas de simplificación el factor de expansión es mucho menor. De esta forma se pueden obtener subgrupos de tamaños más pequeños y comprobar la pérdida de eficacia frente a la ganancia en coste computacional.

Aleatoriedad frente a reducción lineal

Las siguientes figuras ilustran los resultados obtenidos de ejecutar el algoritmo con subgrupos de diferentes densidades (representado en el eje de abscisas), obtenidos respectivamente limitando la longitud y de forma aleatoria a partir del subgrupo más grande generado. Esta vez las pruebas fueron realizadas con el árbol de búsqueda para agilizar el proceso.

En primer lugar, el tiempo (Figura 5). A medida que aumenta la profundidad de recursión desde n=1 hasta n=7, el coste computacional aumenta exponencialmente, dado que las líneas están distribuidas uniformemente a lo largo del eje de ordenadas en escala logarítmica, mientras que a medida que aumentamos la densidad del grupo, el aumento en el coste es mínimo dado que se trata de un árbol optimizado.

Las líneas en escala de azules representan al uso de un grupo aleatorio y los naranjas al grupo acortado limitando la longitud de las cadenas. Al tener ambos el mismo número de elementos, no supone apenas diferencia en coste más allá de la dada por el procesador. A simple vista, la reducción aleatoria parece presentar mejores resultados, pero con unan diferencia despreciable.

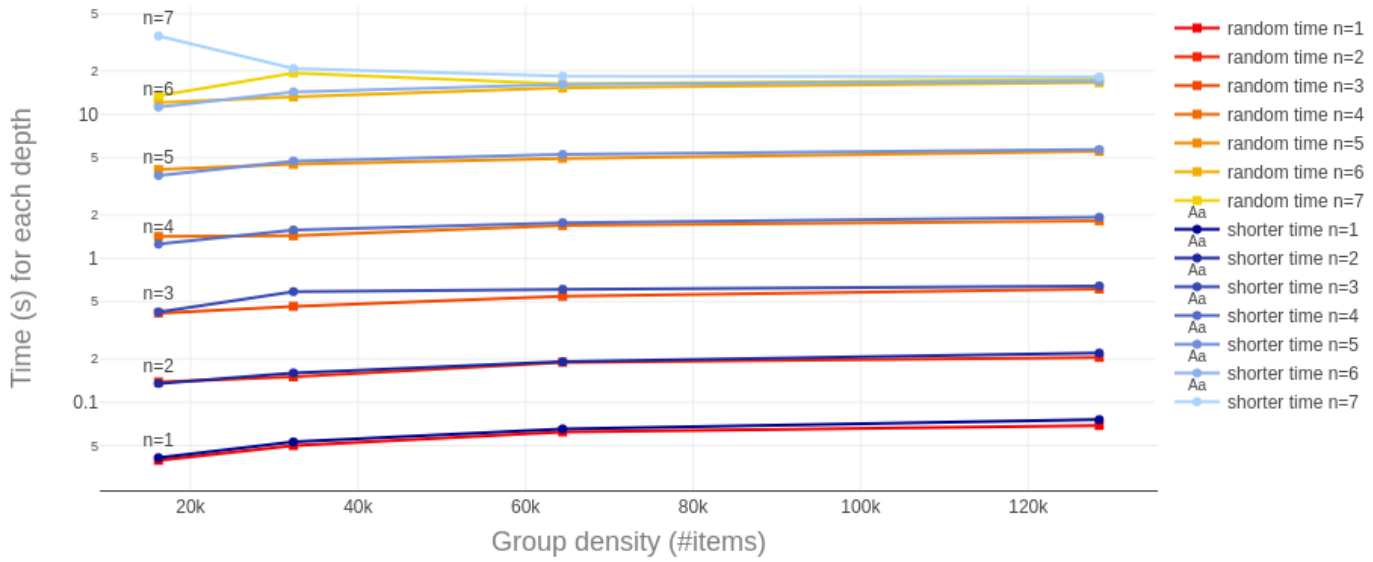

Figura 5: Tiempo de ejecución para varios niveles de recursión con subgrupos reducidos aleatoriamente frente grupos acortados.

En el caso de la tasa de error cometido en la aproximación (Figura 6), se observan diversos picos e incluso un resultado que la intuición podría considerar anormal: esto es, que el resultado para n=7 sea peor que para n=6 para el caso del grupo acortado no aleatoriamente (en escala verde), frente al grupo aleatorio (en escala rosa), que presenta unos resultados más uniformes.

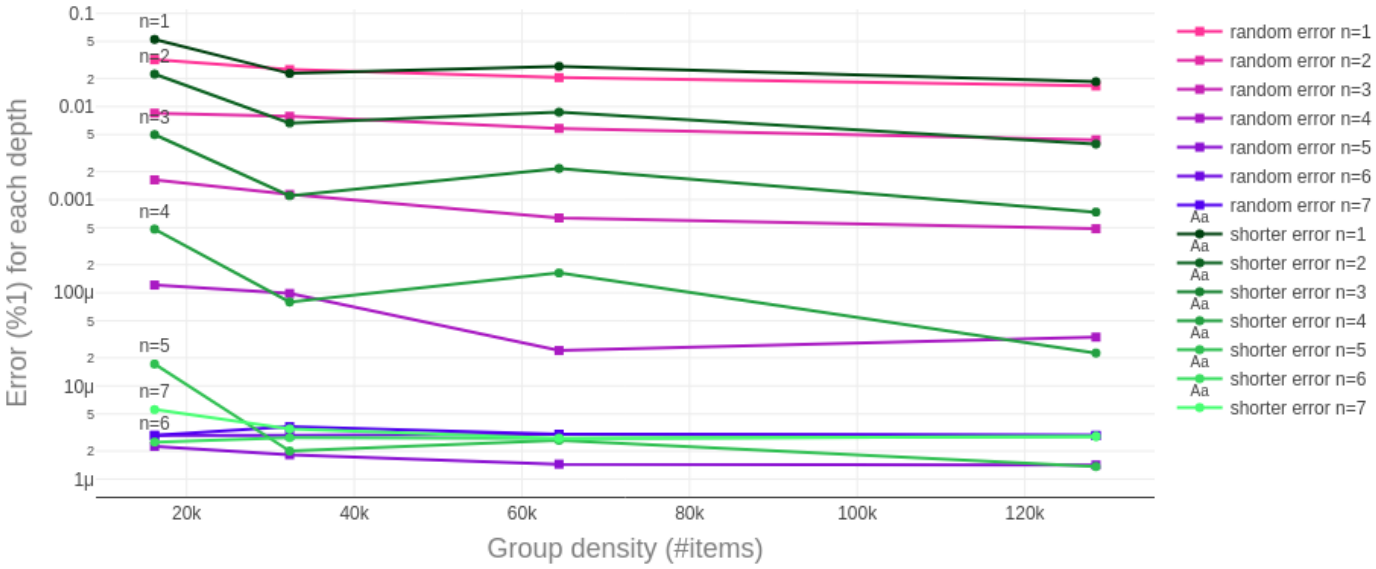

Figura 6: Error obtenido para varios niveles de recursión con subgrupos reducidos aleatoriamente frente grupos acortados.

Estos resultados nos llevan a pensar que no existe una diferencia notable, ni en eficacia ni en eficiencia, entre estos dos criterios de selección de elementos para conformar grupos más cortos para el caso de la búsqueda optimizada.

Reducción con búsqueda optimizada

La búsqueda optimizada permite encontrar vecinos próximos rápidamente para así descartarlos a la hora de generar un grupo más corto, con el inconveniente de un coste añadido de $O(l^2)$, donde l se refiere

al tamaño del subgrupo. La Figura 7 combina los resultados con las tres formas de reducir el grupo (en azul, rojo y verde) para cinco densidades diferentes (de oscuro a claro). Al haber tantas líneas no se aprecia casi nada con claridad, aunque sí se nota que las verdes se encuentran más a la izquierda por tratarse del árbol, y las tres más oscuras más arriba por tratarse de un grupo de sólo 9 elementos, lo que aumenta las probabilidades de que siempre encuentre la misma matriz y por eso el error no varía, además de ser alto.

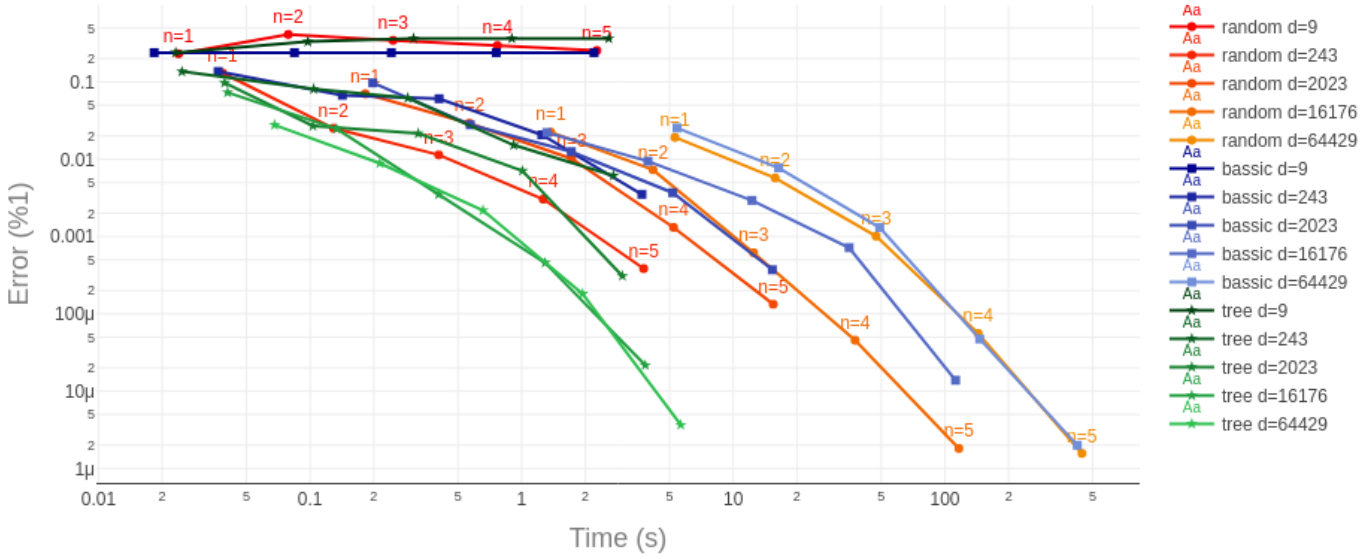

Figura 7: Error frente a tiempo para varios aproximadores, varias densidades de subgrupo y varios niveles de profundidad

Estudiando cada característica por separado y manteniendo el rango de valores en las figuras se aprecia mejor que las verdes (Figura 8) se encuentran más a la izquierda y van bajando a medida que aumenta el tamaño del grupo. Las otras dos se comportan de forma muy similar (Figuras 9 y 10), con un tiempo bajo y error alto para el grupo de 9 elementos y mejorando en eficacia a medida que aumenta el tamaño. Como también se comprobó en el apartado anterior, el grupo aleatorio presenta un error ligeramente menor.

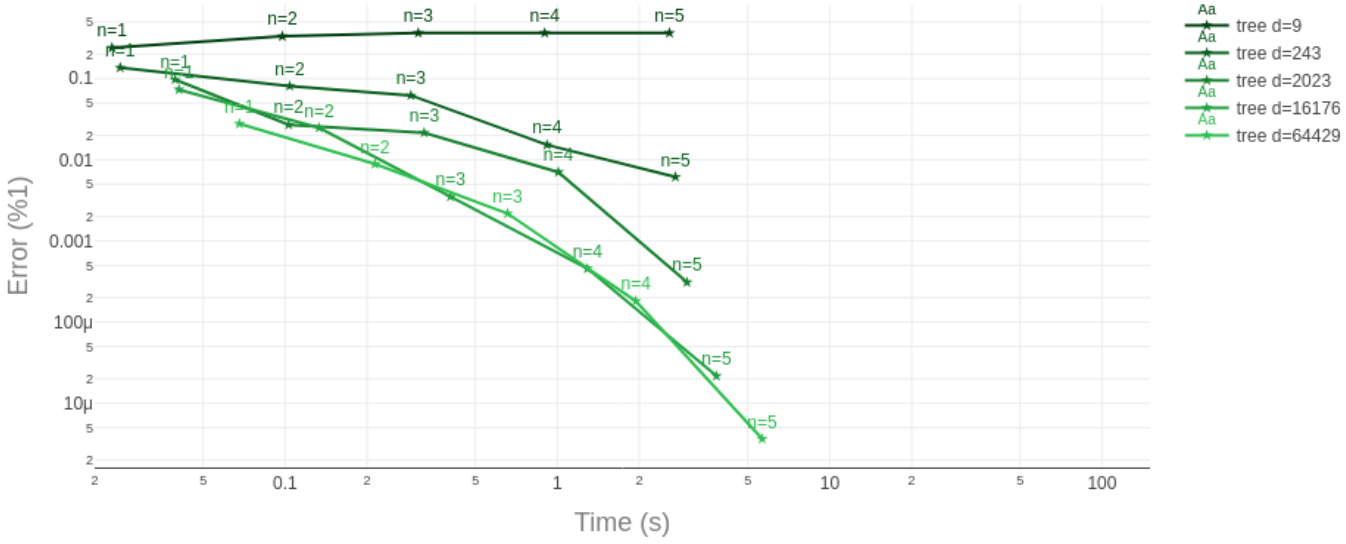

Figura 8: Error frente a tiempo para búsqueda en árbol, varias densidades de subgrupo en varios niveles de profundidad

Las Figuras de la 24 a la 28 (Anexo F) estudian cada densidad por separado. Los picos en d=9 son de esperar, pues un conjunto de 9 operadores no permite sacar conclusiones útiles. A medida que aumenta la densidad, el tiempo para la gráfica verde se reduce, separándose hacia la izquierda de las otras dos mientras que la naranja alcanza límites inferiores mayores indicando, de nuevo, una mejor aproximación para el grupo aleatorio.

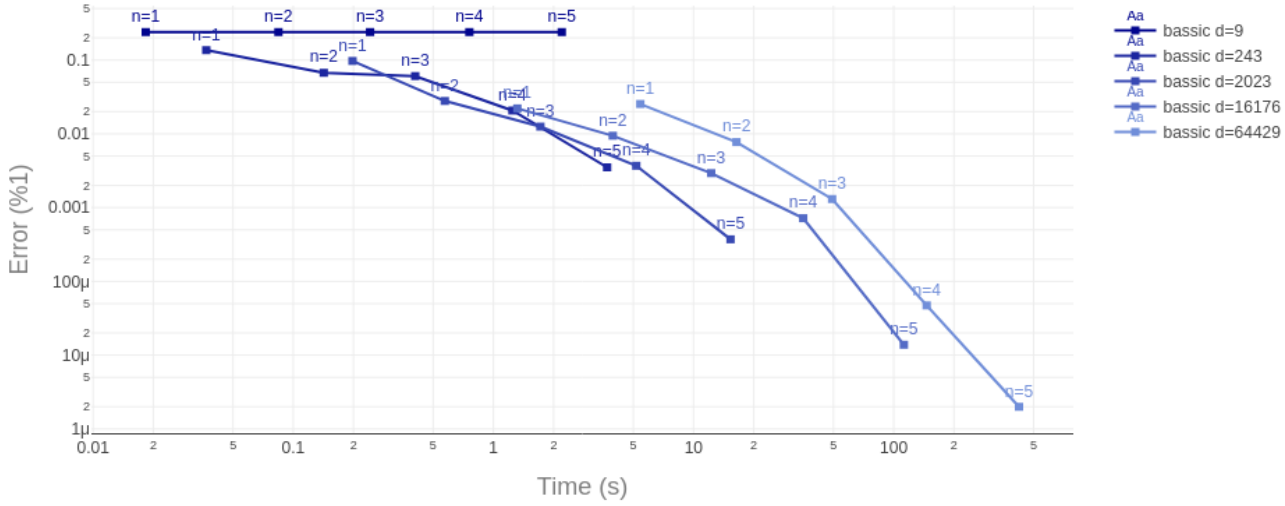

Figura 9: Error frente a tiempo para buscador básico, varias densidades de subgrupo en varios niveles de profundidad

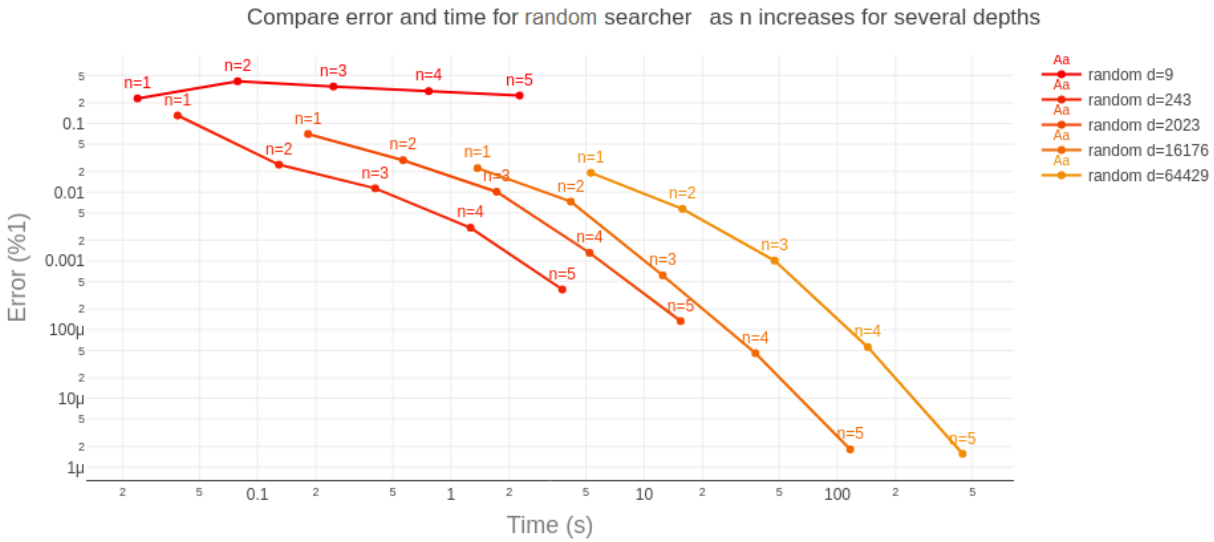

Figura 10: Error frente a tiempo para buscador básico, subgrupos reducidos aleatoriamente en varios niveles de profundidad

Coste frente a eficacia para reducción lineal

La primera pregunta que surge al tomar un grupo menos denso es si el tiempo ganado compensará a la eficacia perdida. Para poder medir esta propiedad no basta con visualizar en conjunto los tiempos y

errores de las densidades, como en la Figura 7, pues lo único que se aprecia es un caos de líneas que no aportan la información que necesitamos.

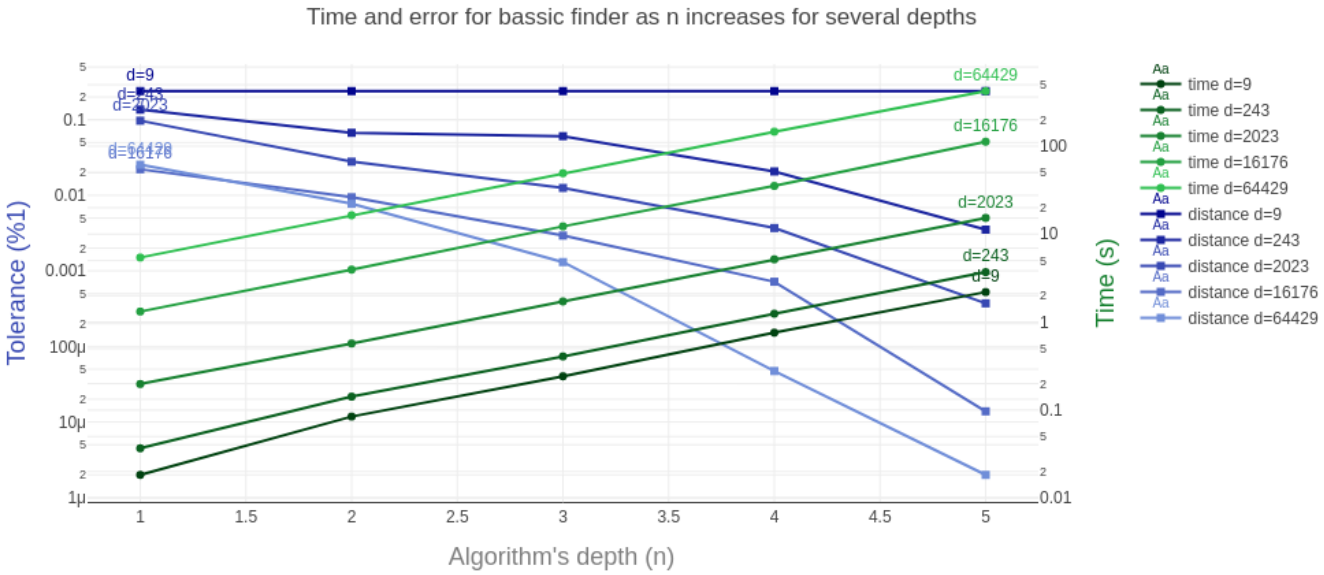

Figura 11: Tiempo y error frente a nivel de profundidad para buscador básico y varias densidades de subgrupo

Llamaremos Tasa a la inversa del producto del tiempo por el error (tolerancia) obtenido para una ejecución en concreto, y tomaremos como referencia la tasa obtenida en el buscador lineal sobre el grupo más grande generado (Figura 12) para poder comparar con ésta las tasas obtenidas al probar con grupos más pequeños.

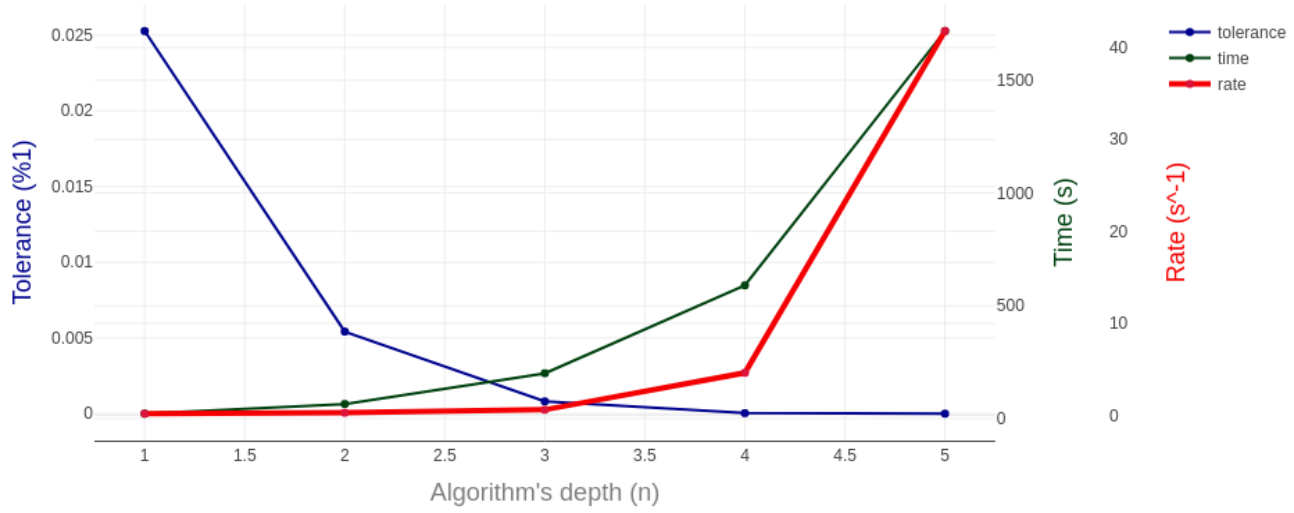

Figura 12: Resultados + Tasa frente a nivel de profundidad para buscador básico y mayor subgrupo

Ya que la tasa se representa en s^{-1}, está representada en un eje aparte. Sabremos si compensa la reducción de tiempo frente al error acumulado si la tasa obtenida supera a los valores de referencia. Si ambas se cortan querrá decir que solo compensa (o deja de compensar) a partir de cierto valor para el nivel de recursión.

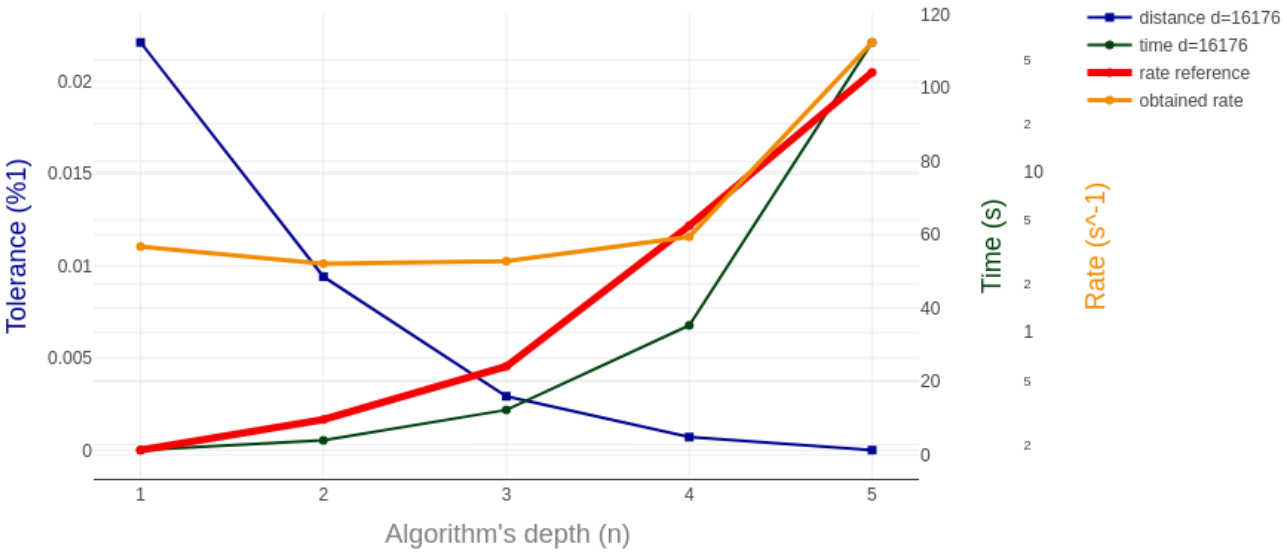

Figura 13: Resultados + Tasa frente a nivel de profundidad para buscador básico y subgrupo de 16176 elementos

Como se aprecia en la Figura 13, para una densidad de aproximadamente ¼ de la original, la línea naranja está siempre por encima de la roja desde n=1, lo cual indica que, para un grupo de esta magnitud, la reducción de tiempo no produce una pérdida de eficacia considerable. En cambio, para las demás (Figuras 29 a 31), la línea naranja deja de estar por encima de la roja a partir de ciertos valores de n, lo cual nos indica que, a partir de cierto nivel de profundidad del algoritmo, es preferible invertir más tiempo encontrar una aproximación mejor. Esto podría variar dependiendo del subgrupo escogido y la matriz que se aproxime, pero vistos estos resultados no hay razones para pensar que cambiar de grupo o de operador suponga una diferencia importante.

5.2 Validación del teorema

Basándonos en los resultados obtenidos empíricamente, el algoritmo de Solovay-Kitaev ha sido capaz de encontrar aproximaciones con un error del orden de 10^{-6} en un tiempo que, si bien resulta exponencial en búsqueda lineal, su crecimiento asintótico es casi nulo si contamos con un árbol de búsqueda previamente calculado y almacenado en memoria.

En esta sección comprobaremos si los postulados mencionados junto con el teorema de Solovay-Kitaev y el pseudocódigo del algoritmo se cumplen para los resultados que ha proporcionado la implementación. Como conclusión a lo visto en la sección 3.3, el teorema afirma que podemos encontrar, para toda puerta U, una ε-aproximación con una longitud de $O(log^{3.97}(1/\varepsilon))$ en un tiempo de $O(log^{2.71}(1/\varepsilon))$.

La longitud de la cadena no ha sido tenida en cuenta a la hora de medir los resultados empíricamente, pues en esta versión del algoritmo solo se ha medido su tiempo de ejecución y lo buena que era la aproximación obtenida. Por este motivo, la validación y comprobación del teorema que se expone a continuación consiste en primer lugar en representar gráficamente las fórmulas recursivas con las que contamos para, posteriormente, comprobar si los resultados empíricos coinciden asintóticamente con los esperados.

5.2.1 Error máximo y nivel de profundidad

Como se menciona en la sección 3.3, estudiando a fondo el pseudocódigo obtenemos la recursión para la calidad de las aproximaciones en cada iteración del algoritmo dada por (9), de la cual se puede deducir (12). Esto nos permite, conocidos los valores de ε_0 y c', conocer el valor de n necesario para obtener una cierta ε-aproximación y, despejando, el mínimo ε que se puede alcanzar con un n fijo.

No obstante, estos valores son puramente teóricos y el resultado puede variar en pruebas empíricas. Para las pruebas se han escogido los valores $c' = 1/32$ y $\varepsilon_0 = 0.014$, que son los que se ofrecen en [14] para los fines prácticos más nos interesan. (En [14] se da un valor de $\varepsilon_0 \approx 0.14$, lo cual ha de ser una errata pues éste ha de ser $< 1/32 = 0.0325$).

Los resultados escogidos provienen de la ejecución del algoritmo con buscadores tanto lineales como con árbol, tomando como subgrupo de $\mathbb{SU}(2)$, por un lado, el de más cardinal, y, en un conjunto aparte, algunos más ligeros. Los puntos del plano en color azul que aparecen representados en la Figura 14 representan para cada error alcanzado, en el eje x, el nivel de profundidad al que se llegó. Los puntos en color verde representan lo mismo, solo que corresponden a resultados obtenidos con subgrupos más pequeños.

Los datos están acompañados de dos gráficas escalonadas que se superponen a ellos en mayor o menor medida. La parte no escalonada corresponde a valores para los que el cálculo del logaritmo se complica: a medida que el valor de ε disminuye, los errores de precisión en coma flotante se disparan, produciendo en ocasiones logaritmos no definidos o divisiones por cero. Esto provocó la imposibilidad de aplicar adecuadamente la función techo ([19]).

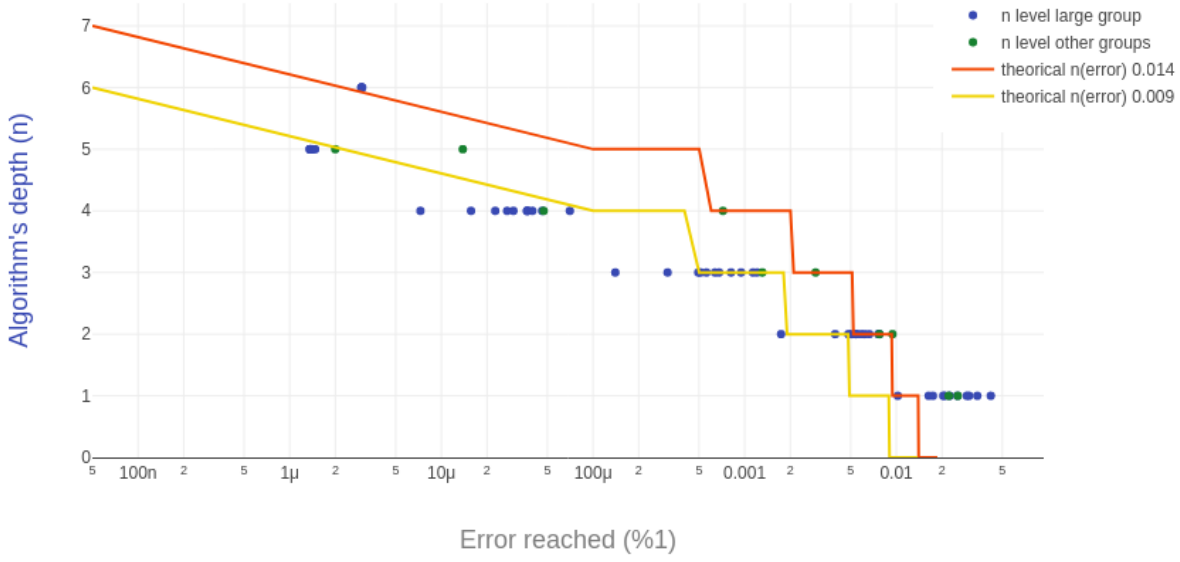

Figura 14: Nivel de recursión requerido según error

Ambas gráficas teóricas están sacadas de la fórmula (12), usando los valores de $\varepsilon_0 = 0.014$ (en rojo) y $\varepsilon_0 = 0.009$ (en amarillo). Cuando los puntos quedan por encima de la gráfica, quiere decir que el nivel de recursión mínimo n requerido ha resultado ser mayor que el valor teórico. Este caso solo se ha dado para n=1 y valores de ε relativamente altos, pues para los demás, los puntos se ajustan con relativa precisión a los escalones de la gráfica.

El hecho de poder "contener" los puntos de datos empíricos entre estas dos "escaleras" permite concluir que ajustando y jugando con los valores de c' y ε_0, se puede llegar a obtener resultados teóricos

que se ajusten a los empíricos. Aunque puede sonar paradójico, en el proceso de búsqueda y delimitación de constantes en fórmulas es común que sean los propios datos experimentales los que permitan ajustarlas.

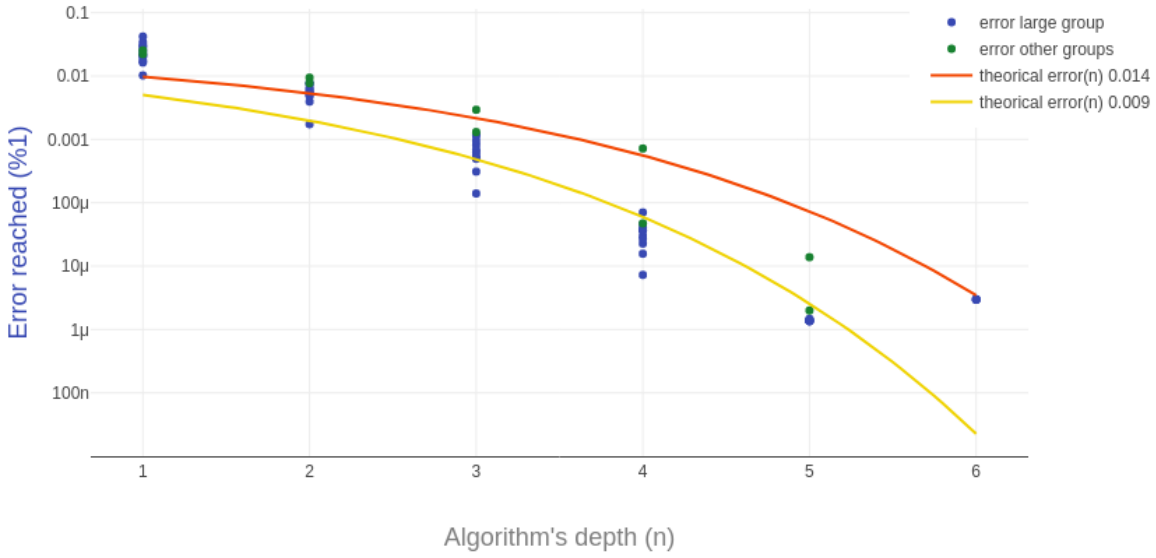

Figura 15: Errores obtenidos para ciertos niveles de recursión

Si representamos ahora el error obtenido dado el nivel de recursión, obtenemos una gráfica similar pero con los ejes invertidos (Figura 15). Al no estar sujetos a tener que tomar la parte entera, las líneas teóricas, en rojo y en amarillo, para los mismos valores de ε_0 que antes, ya no son escalonadas. Vienen dadas por la ecuación (9). Al igual que antes, las gráficas teóricas no se ajustan exactamente pero muestran la misma tendencia asintótica, lo que nos permite concluir que [14] ofrece buenos valores de c' y ε_0 a efectos prácticos.

5.2.2 Tiempo asintótico para error máximo

El objetivo ahora es comprobar si el tiempo de ejecución presenta una tendencia asintótica de $O(log^{2.71}(1/\varepsilon))$, como se deduce de (14). En la Figura 16 se han representado puntos correspondientes al error y tiempo obtenidos de diversas ejecuciones, con diversos tamaños de grupos, que han sido "encerrados" entre dos curvas $y = ln^{2.71}(1/x)$ e $y = 0.003\ ln^{2.71}(1/x)$, lo que prueba que la tendencia asintótica se corresponde en la práctica con las predicciones.

5.3 Conclusiones

El estudio del pseudocódigo y la deducción de las secuencias recursivas y tendencias es una tarea fácil en comparación con la obtención de resultados. Si bien han surgido complicaciones, tanto dudas sobre el dominio como a la hora de enfrentarse a las herramientas, los resultados han resultado ser satisfactorios.

Se ha podido comprobar que la construcción de un buen subgrupo y el análisis de formas de empequeñecerlo para reducir tiempos de ejecución no es una tarea trivial. El uso de un árbol de búsqueda, al igual que para la búsqueda lineal, requiere un previo coste de construcción y guardado en disco. Pero esta tarea solo se realiza una vez y, aunque la construcción de un árbol de búsqueda es ligeramente más costosa, ahorra mucho trabajo al computador y produce los mismos resultados, pues contiene la misma información en sus nodos que si almacenamos el grupo en una lista.

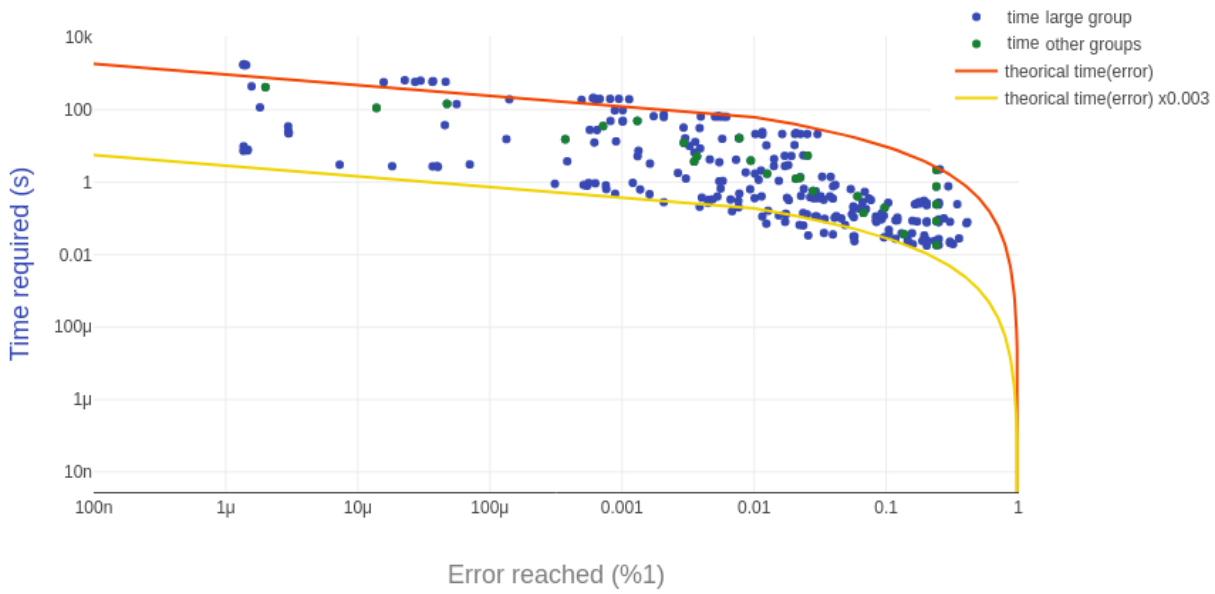

Figura 16: Tiempo requerido para errores dados

El algoritmo de Solovay-Kitaev ha resultado producir resultados satisfactorios para las restricciones impuestas de precisión. Combinando su mecanismo con métodos de búsqueda altamente optimizados y un grupo de partida mucho más denso, sus resultados pueden llegar a ser realmente prometedores para la búsqueda de conjuntos de puertas cuánticas universales (Anexo B).

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Desde sus orígenes, la física es la rama de la ciencia que se ha preocupado por analizar y estudiar el comportamiento del universo, a mayor o menor escala, desde los primeros astrónomos como Galileo Galilei o sus precursores de la Grecia clásica, hasta los últimos siglos, cuando los avances en la ciencia y el descubrimiento de los átomos y las partículas que los componen han abierto una gran cantidad de paradigmas y teorías que explorar.

Ya antes de los inicios de la sociedad clásica, el ser humano ha estado invadido por la necesidad de dar explicación a todo aquello que le rodea. Irónicamente, gracias a la ciencia la humanidad ha podido convencerse de que hay cosas que ni la propia ciencia puede explicar. No obstante, estamos sujetos a una sociedad que se encuentra sometida a un crecimiento exponencial en todos sus aspectos, y puede llegar un momento en el que seamos capaces de desafiar las leyes de la naturaleza.

A pesar de que los primeros estudios de mecánica cuántica datan de hace más de 100 años, su puesta en práctica es un paradigma que se encuentra aún en proceso de desarrollo. Si bien existen teorías de que la física cuántica es "real", y de que el entrelazamiento existe, estas teorías pueden perder su fuerza en el mundo real debido a las limitaciones a las que estamos sujetos los seres humanos. El poder tener control sobre el mundo a escalas pequeñas requiere una tecnología e infraestructuras con las que no podemos contar tan fácilmente como nos gustaría, y es por esto por lo que primero se deben superar estas barreras para poder progresar.

6.2 Trabajo futuro

La primera limitación que presenta este trabajo es el hecho de que todo el desarrollo está adaptado para sistemas de un único *qbit*. El código fuente del que parte la implementación del algoritmo incluía archivos de configuración para preparar la ejecución para sistemas de dos *qbits*, por lo que puede parecer relativamente sencillo implementar la generación del grupo $\mathbb{su}(4)$ y ejecutar pruebas para determinar cuánto coste adicional supone trabajar con operadores de 4×4 o incluso $2^N\times2^N$.

Una de las posibles mejoras que se le puede aplicar al algoritmo es encontrar una forma de sortear el constante cambio de contexto durante la recursión para poder implementar una segunda versión del algoritmo mejor. De esta manera aseguraríamos siempre que la tolerancia alcanzada es la deseada, en lugar de tener que calcular el nivel de profundidad necesario a partir de (12).

Durante la obtención de resultados no se han tomado muestras de la longitud de la cadena de operadores que determinaba la mejor aproximación a pesar de que esta información se almacenaba como una propiedad del valor de retorno. Poder dar soporte a guardar esta información junto con el tiempo y la precisión sería inmediato y podría generar una retroalimentación interesante y más completa sobre la eficacia del algoritmo.

A causa de la necesidad de tratar con números próximos a cero, en ocasiones las operaciones en coma flotante producían resultados erróneos e indeterminaciones inesperadas como, por ejemplo, una división por cero cuando se establecía un nivel de recursión por debajo de 7. Esto se debe a que en una máquina común, *Python* cuenta con una precisión de 53 bits para almacenar números en coma flotante, y pueden no ser suficientes para números decimales que resultan ser periódicos expresados en binario, como por ejemplo $1/10$ ([20]). Traducir la implementación a un lenguaje no interpretado que cuente con más bits

de precisión para operaciones en coma flotante podría proporcionar más libertad para ejecutar el algoritmo a más profundidad y obtener mejores aproximaciones.

Por último, una buena elección del subgrupo de partida puede ser decisivo a la hora de obtener una buena aproximación. Un estudio y análisis del grupo $\mathbb{SU}(2)$ junto con buenas nociones de topología podrían determinar un mecanismo para poder generar un subgrupo finito lo suficientemente denso para garantizar la existencia de un elemento lo suficientemente próximo a cualquier operador. Esto podría establecer el valor de ε_0 que, mientras sea $< 1/32$, cuanto menor sea, se necesitaría menos nivel de recursión en el algoritmo para obtener una ε-aproximación.

7 Referencias

[1] https://cacm.acm.org/magazines/2019/5/236426-quantum-hype-and-quantum-skepticism/fulltext#FND *último acceso 20/5/2019*

[2] https://www.fayerwayer.com/2018/06/summit-supercomputadora-ibm/ *último acceso 27/5/2019*

[3] Einstein, A.; Podolsky, B.; Rosen, N. "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?". Physical Review 47: 777-780, (1935)

[4] Paul Benioff, "The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines", Journal of Statistical Physics, 22, 563, 1980.

[5] National Academies of Sciences, Engineering, and Medicine, "Quantum Computing: Progress and Prospects". Washington, DC: The National Academies Press, 2019

[6] https://www.muyinteresante.es/tecnologia/articulo/ibm-presenta-el-primer-ordenador-cuantico-comercial-de-la-historia-821547024277 *último acceso 13/6/2019*

[7] David Deutsch and Richard Jozsa, "Rapid solutions of problems by quantum computation". Proceedings of the Royal Society of London A 439: 553, 1992.

[8] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. "Quantum algorithms revisited". Proceedings of the Royal Society of London A 454: 339-354, 1998,

[9] Peter W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer"

[10] W. Shor, Peter, "Scheme for reducing decoherence in quantum computer memory", Physical Reviews A, 1995.

[11] Grover, L.K., "A fast quantum mechanical algorithm for database search", Proceedings, 28th Annual ACM Symposium on the Theory of Computing, p. 212, May 1996.

[12] Schumacher, B, "Quantum coding 51", Physical Review A, pp. 2738-2747, 1995.

[13] Claude Cohen-Tannoudji; Bernard Diu; Franck Laloë. "Quantum Mechanics. vol.1" (3ª edición), p. 898, 1977.

[14] Dawson, Christopher M.; Nielsen, Michael A, "The Solovay-Kitaev algorithm", (2005-05-05)

[15] https://github.com/cryptogoth/skc-python *último acceso 12/6/2019*

[16] https://plot.ly/ *último acceso 15/6/2019*

[17] https://docs.scipy.org/doc/numpy/reference/generated/numpy.ceil.html *último acceso 10/6/2019*

[18] R. P. Feynman, "Quantum mechanical computers", Optics News, 11, p. 11, febrero 1985.

[19] https://docs.scipy.org/doc/numpy/reference/generated/numpy.ceil.html *último acceso 13/6/2019*

[20] https://docs.python.org/2/tutorial/floatingpoint.html *último acceso 15/6/2019*

8 Glosario

FLOPS	Floating point Operations Per Second. Operaciones en coma flotante por segundo.
RSA	Rivest, Shamir y Adleman. Sistema criptográfico.
Qbit	Unidad básica de información en un ordenador cuántico.
Qdit	Sistema de d *qbits*. Vector 2^d-dimensional.
Capítulo	Parte del documento que se encuentra bajo un título de nivel 1.
Sección	Parte del documento que se encuentra bajo un título de nivel 2.
Subsección	Parte del documento que se encuentra bajo un título de nivel 3.
Apartado	Parte del documento que se encuentra bajo un título de nivel 4.
BSP	Binary Space Partitioning. Partición binaria del espacio.
Subgrupo	Subconjunto de un grupo algebraico que preserva las propiedades y tiene por tanto identidad propia de grupo independente de su contenedor.
Shell	Consola de commandos.
Script	Archivo de órdenes.
Polimorfismo	Propiedad de la orientación a objetos por la que es posible ejecutar órdenes distintas pero sintácticamente idénticas.
$\|\|\cdot\|\|$	Norma vectorial.
$\|\cdot\|$	Módulo complejo.

9 Anexos

A De la función de onda a la ecuación de Schrödinger

Una función de onda se utiliza para representar en qué estado físico se encuentra un sistema de partículas. Se denota por ψ(x; t), donde x representa la posición, un vector que puede ser complejo, y t el tiempo. Tiene la particularidad de ser una función de cuadrado integrable, es decir que la integral de su cuadrado es un valor finito, lo que implica que es medible dentro de un espacio de Hilbert.

Fue junto con el auge de la primera física cuántica cuando comenzó a interpretarse que la posición y propagación de las partículas en el espacio podía ser determinada por una función. El concepto evolucionó y, en formulación moderna, la función de onda es un objecto más abstracto que permite representar y agrupar los posibles estados de un sistema como elementos de un espacio de Hilbert.

La ecuación de Schrödinger fue descrita en 1925 por el físico austriaco Erwin Schrödinger. Representa un papel análogo a la segunda ley de Newton (la aceleración de un cuerpo se obtiene del cociente de la suma de sus fuerzas por su masa) para sistemas descritos en el marco de la mecánica cuántica, y explica la evolución temporal de una función de onda.

La serie de sucesos que llevaron a Schrödinger a postular la ecuación partieron del ya conocido a principios del siglo XX hecho de la dualidad onda-corpúsculo de la luz. Esto es que la luz puede comportarse tanto como una onda electromagnética como una partícula (un fotón). El trabajo de Schrödinger estuvo basado en las investigaciones previas de otros científicos de la época como Louis-Victor de Broglie y los experimentos realizados por Clinton Davisson y Lester Germer.

Davisson y Germer comprobaron que la hipótesis de de Broglie de 1923 de que a toda partícula clásica puede asociársele una función de onda era cierta. De Broglie asoció a cada partícula libre con energía E y cantidad de movimiento p una frecuencia ν y una longitud de onda λ (ecuación (17)). Esta hipótesis fue también lo que inspiró a Schrödinger a determinar una ecuación, para la onda de de Broglie para escalas microscópicas, pero que a escalas macroscópicas fuese equivalente a la ecuación de la partícula en mecánica clásica. Resultó en la ecuación (18).

$$\begin{cases} E = h\nu \\ p = {}^{h}/_{\lambda} \end{cases} \quad \text{siendo} \quad \begin{cases} p = momento\ lineal \\ h = cte\ de\ Plank \\ \quad = 6{,}626 \cdot 10^{-34}\ kg \cdot m^2 \cdot s^{-1} \end{cases} \tag{17}$$

$$E = \frac{p^2}{2m} + V(r) \quad \text{siendo} \quad \begin{cases} m = masa \\ p = momento\ lineal \\ V = energía\ potencial \\ r = posición \end{cases} \tag{18}$$

A pesar del hecho de que Max Born diera en 1926 una interpretación física válida de la ecuación, otros físicos como Albert Einstein manifestaron su desconfianza ante ella a causa del carácter probabilístico que inducía, llegando incluso a suscitar dudas en el propio Schrödinger.

A principios de la década siguiente, Born dio a la ecuación de Schrödinger una interpretación probabilística basada en el hecho de que la versión compleja de la ecuación tiene una integral de movimiento que puede ser interpretada como una densidad de probabilidad. Así la función de onda pasó a interpretarse de modo más abstracto como una amplitud de probabilidad en lugar de ser enfocada como una onda material.

En mecánica cuática, esta interpretación probabilística permite definir el estado de un sistema como un vector unitario (unívocamente determinado) en un espacio de Hilbert complejo y separable, el cual además representa el conjunto de todos los estados posibles de dicho sistema. Como un sistema evoluciona con el tiempo, y además depende de la posición, el vector unitario que determina su estado ha de estar también en función del tiempo y de la posición.

Finalmente, la formulación moderna de la ecuación de Schrödinger viene dada por (19). Combina los postulados del físico con la notación *bra-ket* de Paul Dirac que representan sistemas cuánticos. Gracias a esta formulación, que determina la evolución temporal de $|\varphi(t)\rangle$, $\hat{H}(|\varphi(t)\rangle)$ (llamado Hamiltoniano, el observable que representa la energía total del sistema), pueden obtenerse las probabilidades de todos los posibles resultados de las medidas.

$$\widehat{H}(|\varphi(t)\rangle) = i\frac{h}{2\pi}|\varphi(t)\rangle = \frac{\hat{\vec{p}}^2}{2m}|\varphi(t)\rangle + V(\hat{\vec{r}},t)|\varphi(t)\rangle \qquad (19)$$

Donde $\begin{cases} i: unidad\ imaginaria \\ \hat{\vec{p}}: vector\ momento\ lineal \\ \hat{\vec{r}}: vector\ posición \\ h: cte\ de\ Plank \\ V: energía\ potencial \end{cases}$

No obstante, la ecuación presenta algunas limitaciones. En primer lugar sólo es válida para describir partículas cuyo momento lineal sea pequeño en comparación con la energía entre la velocidad de la luz. Esto es porque se trata de una ecuación no relativista. Además, no incorpora adecuadamente el *spin* de las partículas. Paul Dirac y Pauli proporcionaron más tarde la ecuación de Dirac, consiguiendo con ella resolver este problema e incorporar o solo los efectos del *spin* sino también efectos relativistas.

B Puertas y circuitos cuánticos

Un circuito cuántico está formado por un conjunto de objetos llamados puertas cuánticas. La notación actual para las puertas y circuitos cuánticos se atribuye originalmente a Richard Phillips Feynman ([18]).

Una puerta cuántica es un operador actúa sobre sistemas de N *qbits*, los transforma y devuelve como resultado otro sistema también compuesto de N *qbits*. La principal diferencia entre un circuito clásico y uno cuántico reside en que las puertas cuánticas, al contrario que las clásicas, son reversibles. Esto quiere decir que dado el valor de salida de una puerta cuántica, podemos conocer el valor de partida, lo que permite tener un control mucho más amplio sobre el comportamiento de los circuitos.

La razón por la que una puerta cuántica es reversible es que se representan mediante matrices unitarias, de dimensión $2^N \times 2^N$, ya que los *qbits* presentan una naturaleza que permite tratarlos como vectores y, si una matriz unitaria se aplica sobre un vector, se puede regresar al vector de partida invirtiendo la matriz.

Al igual que en los circuitos clásicos, las puertas cuánticas cuentan con una notación particular esquemática. A continuación se ilustran los ejemplos más conocidos en la actualidad.

Puerta Hadamard

La puerta Hadamard actúa sobre un único *qbit*, por lo que se representa mediante una matriz 2×2. También es la representación en un *qbit* de la transformada cuántica de Fourier, una transformación análoga a la transformada de Fourier discreta. La Tabla 1 ilustra la salida de la puerta Hadamard dados los estados puros $|0\rangle$ y $|1\rangle$. Corresponde a la matriz de la fórmula (20). La Figura 17 es la representación esquemática dentro de un circuito.

$$\mathrm{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{20}$$

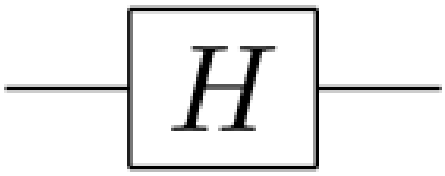

Figura 17: Representación de la puerta Hadamard en un circuito

Puerta SWAP

La puerta SWAP actúa sobre dos *qbits* y los intercambia (SWAP, del inglés cambiar). Como se ilustra en la Tabla 3, deja intactos los estados entrelazados $|00\rangle$ y $|11\rangle$ e invierte a $|01\rangle$ y $|10\rangle$. Se representa por la matriz 4×4 (21), y su representación esquemática se ilustra en la Figura 18.

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{21}$$

Qbits entrada	Salida
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|10\rangle$
$\|10\rangle$	$\|01\rangle$
$\|11\rangle$	$\|11\rangle$

Tabla 3: Puerta SWAP

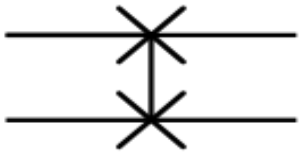

Figura 18: Representación de la puerta SWAP en un circuito

Puertas de desplazamiento de fase

La familia de puertas de desplazamiento de fase, Rϕ, operan sobre un único *qbit*. Dejan intacto al estado básico $|0\rangle$ y asignan a $|1\rangle$ el estado $e^{i\phi}|1\rangle$ (Tabla 4). Aunque añaden información de fase, la probabilidad de obtener $|0\rangle$ o $|1\rangle$ no se ve alterada (ver ecuación (22)). Destacan algunos casos particulares como la puerta T ($\phi = {}^{\pi}/_{4}$) la puerta S ($\phi = {}^{\pi}/_{2}$) y la puerta Pauli-Z ($\phi = \pi$)

Qbit entrada	Salida
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$e^{i\phi}\|1\rangle$
$\alpha\|0\rangle + \beta\|1\rangle$	$\alpha\|0\rangle + \beta e^{i\phi}\|1\rangle$

Tabla 4: Puerta Rϕ

$$R_\phi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \Rightarrow$$

$$R_\phi|\varphi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta e^{i\phi} \end{pmatrix} \Rightarrow \begin{cases} \langle\varphi|0\rangle = |\alpha|^2 = \langle R_\phi\varphi|0\rangle \\ \langle\varphi|1\rangle = |\beta|^2 \\ \qquad\quad = \left|e^{i\phi}\beta\right|^2 = \langle R_\phi\varphi|1\rangle \end{cases} \tag{22}$$

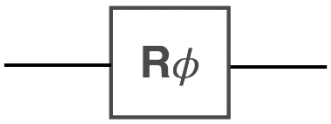

Figura 19: Representación de la puerta Rϕ en un circuito

Puertas de Pauli

Actúan sobre un único *qbit* y su representación es mediante las denominadas matrices de Pauli, que se estudian con más detalle en el Anexo D. Las puertas de Pauli son 3, una por cada eje cartesiano, y equivalen a un giro de π radianes sobre dichos ejes. La Tabla 5 ilustra los valores que genera sobre los estados básicos, y la Figura 20 su representación esquemática. Las marices por las que se representan aparecen en el Anexo D, denotadas por σ_x, σ_y y σ_z. Al tratarse de un giro de π radianes, aplicar estas puertas dos veces seguidas resultaría en el estado de origen. Por eso se dice que son involuciones: sus matrices al cuadrado resultan en la identidad.

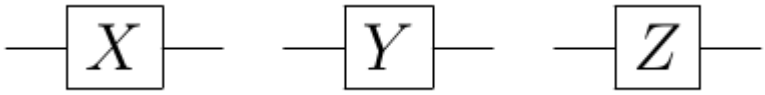

Figura 20: Representación de las puertas de Pauli en un circuito

Qbit entrada	Salida		
	Pauli-X (NOT)	Pauli-Y	Pauli-Z (R_π)
$\|0\rangle$	$\|1\rangle$	$i\|1\rangle$	$\|0\rangle$
$\|1\rangle$	$\|0\rangle$	$-i\|0\rangle$	$-\|1\rangle$

Tabla 5: Puertas de Pauli

Figura 21:Representación alternativa de la puerta Pauli-X / NOT

Puertas controladas

Una puerta controlada se construye a partir de una puerta arbitraria U, que actúa sobre N *qbits* y añadiendo K *qbits* de control, resultando en una puerta que actúa sobre N+K *qbits* representada por una matriz de $2^{N+K} \times 2^{N+K}$. Estas puertas no alteran la entrada cuando los *qbits* de control son 0, y aplican la puerta U cuando son 1. Los *qbits* de control siempre se mantienen inalterados. Suponiendo un único *qbit* de control, su representación matricial es

$$C(U) = \begin{pmatrix} 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 0 & u_{11} & u_{12} \\ \vdots & \vdots & u_{21} & \ddots \end{pmatrix}$$

Cuando existen K *qbits* de control y U actúa sobre N *qbits*, resulta

$$C(U) = \begin{pmatrix} Id_{2^{N+K}-2^N} & 0 \\ 0 & U \end{pmatrix} \in M_{2^{N+K} \times 2^{N+K}}(\mathbb{C})$$

Qbits control	Entrada	Salida
$\|00 \ldots 0\rangle$	$\|\varphi\rangle$	$\|\varphi\rangle$
$\|00 \ldots 1\rangle$	$\|\varphi\rangle$	$\|\varphi\rangle$
…	…	…
$\|11 \ldots 0\rangle$	$\|\varphi\rangle$	$\|\varphi\rangle$
$\|11 \ldots 1\rangle$	$\|\varphi\rangle$	$U\ (\|\varphi\rangle)$

Tabla 6: Puerta genérica U controlada con K qbits de control

Las fórmulas (23) y (24) ilustran ejemplos de las puertas controladas más conocidas, la puerta NOT controlada (o CNOT) y la puerta Toffoli (CCNOT), respectivamente. Los valores para la puerta CNOT vienen dados por la Tabla 1.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{23}$$

Qbits control	Entrada	Salida
$\|00\rangle$	$\|0\rangle$	$\|0\rangle$
$\|00\rangle$	$\|1\rangle$	$\|1\rangle$
$\|01\rangle$	$\|0\rangle$	$\|0\rangle$
$\|01\rangle$	$\|1\rangle$	$\|1\rangle$
$\|10\rangle$	$\|0\rangle$	$\|0\rangle$
$\|10\rangle$	$\|1\rangle$	$\|1\rangle$
$\|11\rangle$	$\|0\rangle$	$\|1\rangle$
$\|11\rangle$	$\|1\rangle$	$\|0\rangle$

Tabla 7: Puerta Toffoli

$$Toffoli\ /\ CCNOT = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{24}$$

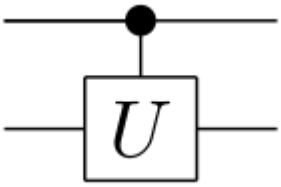

Figura 22: Representación de la puerta genérica U controlada en circuito

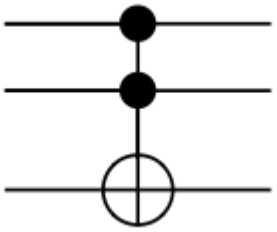

Figura 23: Representación de la puerta Toffoli en circuito

Puertas universales

Un conjunto de puertas universales es aquel a partir del cual se puede construir cualquier puerta cuántica deseada por medio de una combinación finita de puertas del conjunto. A priori es imposible la existencia de tal conjunto, pues si bien éste ha de ser finito, existe un número infinito de posibles puertas cuánticas. Como solución a este problema se propone obtener no cualquier puerta deseada, sino una aproximación lo suficientemente buena como se desee. Teoremas como el de Solovay-Kitaev a firman que es posible encontrar dicha secuencia, para cualquier puerta deseada, a partir de un conjunto finito original que, como se estudia en este trabajo, es un subgrupo denso del grupo unitario especial $\mathbb{SU}(2)$.

C Invarianza respecto a la fase global

Un *qbit* no tiene una única representación. Puede verse o bien como un punto en la superficie de una esfera, o como una combinación lineal de los estados puros $|0\rangle$ y $|1\rangle$: $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$, donde α y β son números complejos ligados por la ecuación (1).

Un número complejo z se puede representar en forma binómica (25), polar (26), o exponencial (27):

$$z = a + bi \tag{25}$$

$$z = |z|_\theta \tag{26}$$

$$z = re^{i\phi} = r\cos(\phi) + i\, r\, sen(\phi) \tag{27}$$

Si escribimos α y β en forma exponencial, tenemos un *qbit* representado de la siguiente forma:

$$|\varphi\rangle = r_\alpha e^{i\phi_\alpha}|0\rangle + r_\beta e^{i\phi_\beta}|1\rangle \tag{28}$$

Así el *qbit* queda determinado por 4 parámetros reales r_α, r_β, ϕ_α y ϕ_β. No obstante, las únicas cantidades medibles que proporcionan información del *qbit* son $|\alpha|^2$ y $|\beta|^2$, y si multiplicamos el estado z por un número complejo arbitrario de norma 1 $e^{i\gamma}$ (una fase global), estos factores no se ven afectados:

$$\left|e^{i\gamma}z\right|^2 = \overline{(e^{i\gamma}z)}(e^{i\gamma}z) = (e^{-i\gamma}\bar{z})(e^{i\gamma}z) = \bar{z}z = |z|^2$$

Esto se conoce como invarianza respecto a la fase global. Aplicando esto al estado $|\varphi\rangle$, sea

$$|\varphi'\rangle = e^{-i\phi_\alpha}|\varphi\rangle = r_\alpha e^{-i\phi_\alpha}e^{i\phi_\alpha}|0\rangle + r_\beta e^{-i\phi_\alpha}e^{i\phi_\beta}|1\rangle = r_\alpha|0\rangle + r_\beta e^{i(\phi_\beta - \phi_\alpha)}|1\rangle = r_\alpha|0\rangle + r_\beta e^{i\phi}|1\rangle$$

Donde $\phi = \phi_\beta - \phi_\alpha$, lo que indica que, efectivamente, a la hora de medir un *qbit* la información de la fase global queda determinada por la diferencia entre ambas. Así el *qbit* $|\varphi'\rangle$ queda determinado por 3 parámetros reales, siendo el coeficiente de $|0\rangle$ un número real no negativo, y conserva los mismos valores de probabilidad $|\alpha|^2$ y $|\beta|^2$ que $|\varphi\rangle$ ($r_\alpha{}^2$ y $r_\beta{}^2$). Si lo combinamos ahora con la restricción dada por (1) y escribimos $r_\beta e^{i\phi}$ en forma binómica, resulta:

$$r_\beta e^{i\phi} = x + iy \tag{29}$$

$$\langle\varphi|\varphi\rangle = 1 \Leftrightarrow \left|r_\alpha|0\rangle + r_\beta e^{i\phi}|1\rangle\right| = 1 \Leftrightarrow \tag{30}$$

$$r_\alpha{}^2 + |x + iy|^2 = 1 \Leftrightarrow r_\alpha{}^2 + x^2 + y^2 = 1$$

Lo que nos deja con una ecuación equivalente a un punto sobre la superficie de una esfera tridimensional de radio 1. Esto nos proporciona una forma de representar el *qbit* en coordenadas cartesianas en $\mathbb{R}^3$. Si un punto sobre la esfera se representa en términos de dos ángulos θ y ϕ:

$$\begin{cases} x = \cos(\phi)\, sen(\theta) \\ y = sen(\phi) sen(\theta) \\ z = r_\alpha = \cos(\theta) \end{cases} \qquad 0 \le \theta \le \pi,\ 0 \le \phi < 2\pi$$

Podemos combinar esto con (28) y (29) para obtener

$$|\varphi\rangle = r_\alpha|0\rangle + (x + iy)|1\rangle = \cos(\theta)\,|0\rangle + \big(\cos(\phi)\,sen(\theta) + isen(\phi)sen(\theta)\big)|1\rangle$$
$$= \cos(\theta)\,|0\rangle + sen(\theta)e^{i\phi}|1\rangle$$

Con esta configuración, para $\theta = \pi/2$ se tiene que $|\varphi\rangle = |1\rangle$, y si $\theta = 0$, $|\varphi\rangle = |0\rangle$, lo que sugiere que para determinar todos los estados basta con $0 \leq \theta \leq \frac{\pi}{2}$. Para poder establecer que el estado $|1\rangle$ corresponde al polo sur, cambiamos el parámetro θ por $\theta/2$ y obtenemos la ecuación (5), que además permite obtener una correspondencia directa entre un estado expresado en términos de α y β, de θ y ϕ y en coordenadas esféricas:

$$|\varphi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle$$

D Spin y matrices de Pauli

El 'espín', del inglés *spin* (giro, girar) es una propiedad física intrínseca de las partículas que termina su momento angular. Es una medida cuantizada, es decir, sólo puede tomar valores determinados: un múltiplo entero del valor $^{h}/_{4\pi}$ (donde h es la constante de Planck), y su magnitud total es única para cada tipo de partícula.

Las matrices de Pauli (en honor a Wolfgang Ernst Pauli) son un tipo particular de matrices útiles en mecánica y física cuántica para cuantificar el *spin* de una partícula. Estas matrices constituyen además, en su versión de dimensión 2, una base del grupo especial unitario $\mathbb{su}(2)$. En el caso de *spin* ½ son 3, como la dimensión del grupo $\mathbb{su}(2)$. Comúnmente, en esta versión se representan de la siguiente forma:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Presentan propiedades interesantes como ser matrices de determinante -1, traza 0 y ser raíces de la unidad. También juegan el papel de operadores sobre un espacio de Hilbert de dimensión 2s + 1, con s = *spin* de la partícula, ½ en este caso.

La medida de un *spin* ha de ser tomada en una dirección determinada, obteniendo como valor la proyección del *spin* sobre esa dirección o eje en concreto. Cuando se representa, en mecánica cuántica, el operador de *spin* una partícula de *spin* ½ (como un electrón, protón o neutrón), a lo largo de la dirección k, el valor viene dado por $^{h}/_{4\pi}\,\sigma_k$, siendo σ_k la matriz de Pauli asociada a la dirección k. La medición del *spin* sobre un estado $|\varphi\rangle$, $S(|\varphi\rangle)$ viene determinada por $(S_x, S_y, S_z)|\varphi\rangle$, donde S_x, S_y, S_z son las matrices de Pauli σ_x, σ_y, σ_z expresadas en la base determinada por los autovectores de σ_z, que resulta ser la identidad.

Esto quiere decir que, expresado en términos de una base dada por S_z, es decir, el eje z en coordenadas cartesianas, el vector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ se corresponde con el estado 'arriba', denotado por $|\uparrow\rangle$, el polo norte de la esfera, y el $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ con el estado 'abajo' ($|\downarrow\rangle$).

Dada su naturaleza cuántica, las propiedades del *spin* han resultado ser una interesante fuente de capacidad aprovechable en el desarrollo de computadores cuánticos y para el proceso de reemplazo de bits convencionales por *qbits*.

E Inversa de matrices 2×2

Dada una matriz $A_{n\times n}$ invertible (de determinante no nulo), hallar su inversa consiste en encontrar una matriz $X_{n\times n}$ tal que AX = XA = I_n (la matriz identidad de dimensión n). Una forma intuitiva de hacerlo es planteando y resolviendo por el método de Gauss los n sistemas de n ecuaciones cada uno que resultan de tratar como variables los n^2 coeficientes de X:

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}\begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nn} \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$$

Otra forma es proceder por el método de los adjuntos. Para ello es necesario hallar la matriz adjunta adj(A). Ésta se calcula hallando la matriz de cofactores de la matriz traspuesta A^T (se obtiene intercambiando filas por columnas). El término de la fila i y columna j de la matriz de cofactores de una matriz B se obtiene de la siguiente forma:

$$coff(B)_{i,j} = (-1)^{i+j} \cdot \det \hat{B}(i,j)$$

Donde $\hat{B}(i,j)$ es la matriz de dimensión n-1 resultante de eliminar la fila i y la columna j de B. Es trivial comprobar que trasponer la matriz de cofactores es lo mismo que hallar los cofactores de la matriz traspuesta. Finalmente, la regla de los adjuntos proporciona la siguiente fórmula:

$$A^{-1} = \frac{1}{\det A} coff(A^T) = \frac{1}{\det A}\begin{pmatrix} \det \hat{A}(1,1) & -\det \hat{A}(2,1) & \cdots \\ -\det \hat{A}(1,2) & \det \hat{A}(2,2) & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

Para el caso n=2, aplicar este método resulta en una conocida fórmula muy sencilla:

$$A^{-1} = \begin{pmatrix} \mathrm{a} & \mathrm{b} \\ \mathrm{c} & \mathrm{d} \end{pmatrix}^{-1} = \frac{1}{ad-bc}\begin{pmatrix} \mathrm{d} & -\mathrm{b} \\ -\mathrm{c} & \mathrm{a} \end{pmatrix}$$

F Figuras

Las Figuras de la 24 a la 28 muestran la comparación entre el error obtenido y el tiempo empleado para subgrupos más ligeros que el original, según varios niveles de profundidad.

Las Figuras de la 29 a la 31 representan la tasa obtenida, según la profundidad de recursión, para subgrupos menos densos que el original.

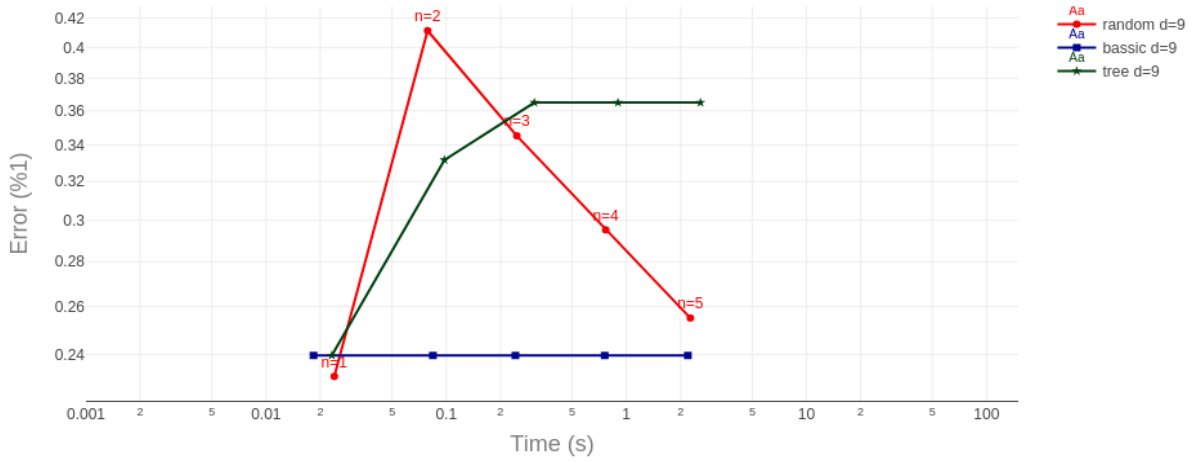

Figura 24: Error frente a tiempo para los tres buscadores, subgrupo de 9 elementos en varios niveles de profundidad

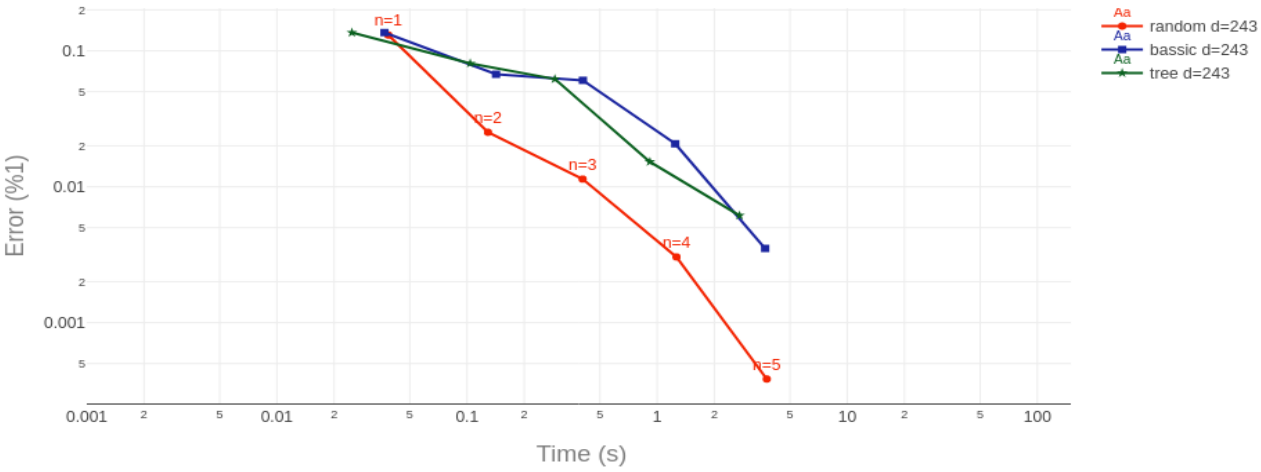

Figura 25: Error frente a tiempo para los tres buscadores, subgrupo de 243 elementos en varios niveles de profundidad

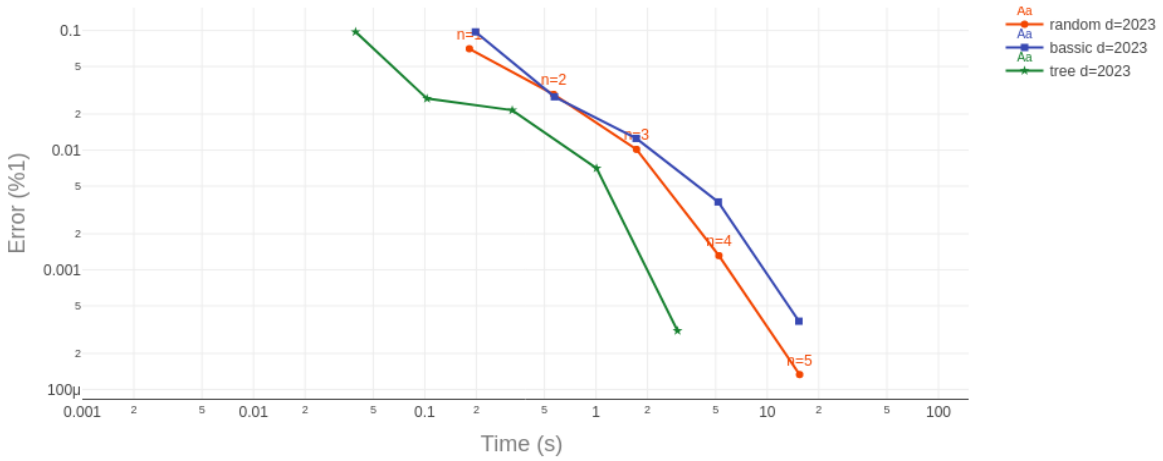

Figura 26: Error frente a tiempo para los tres buscadores, subgrupo de 2023 elementos en varios niveles de profundidad

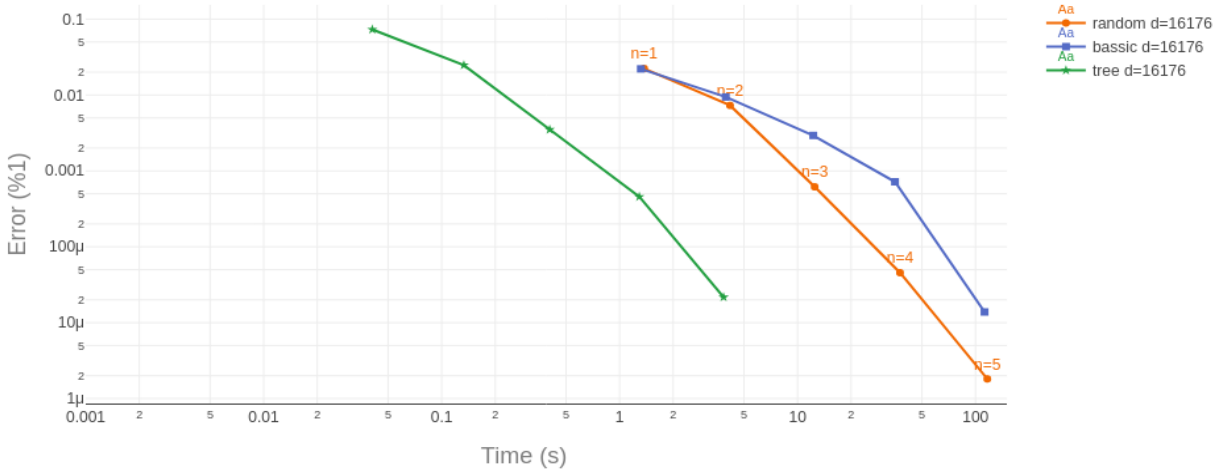

Figura 27: Error frente a tiempo para los tres buscadores, subgrupo de 16176 elementos en varios niveles de profundidad

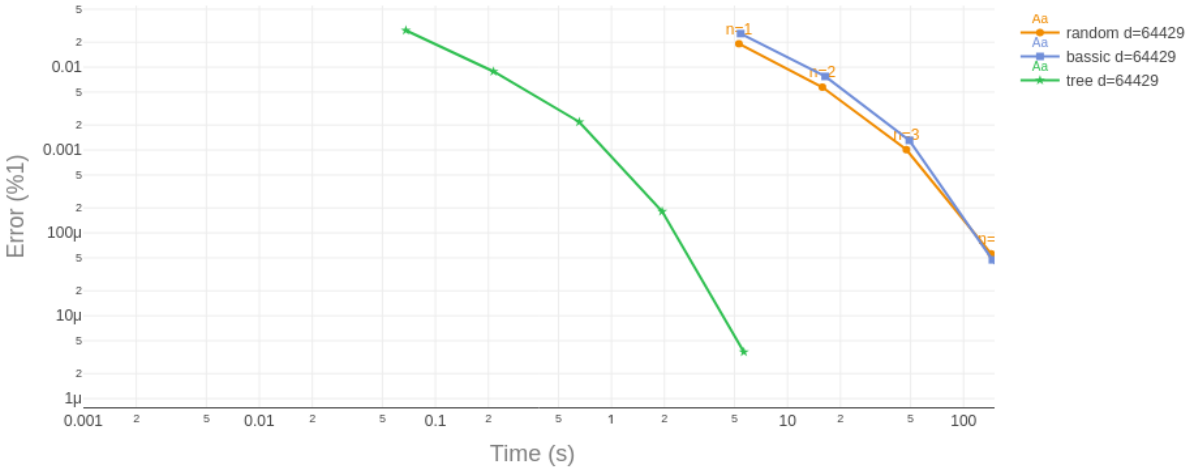

Figura 28: Error frente a tiempo para los tres buscadores, subgrupo de 64429 elementos en varios niveles de profundidad

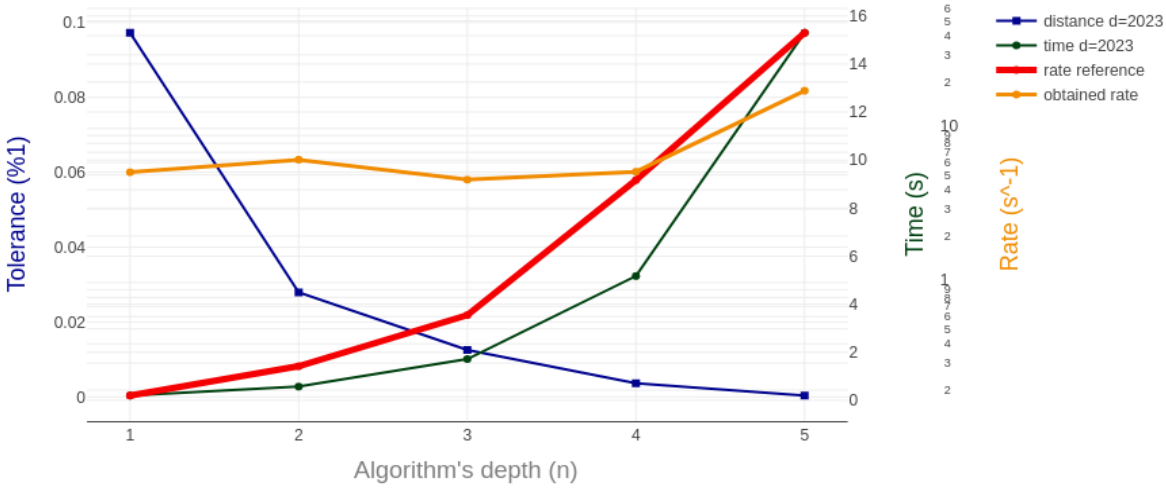

Figura 29: Resultados + Tasa frente a nivel de profundidad para buscador básico y subgrupo de 16176 elementos

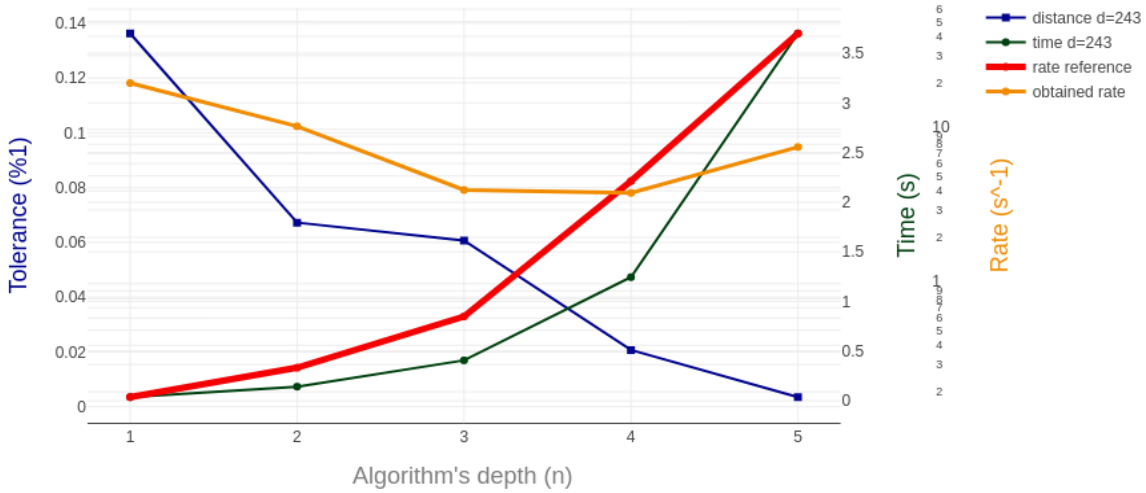

Figura 30: Resultados + Tasa frente a nivel de profundidad para buscador básico y subgrupo de 243 elementos

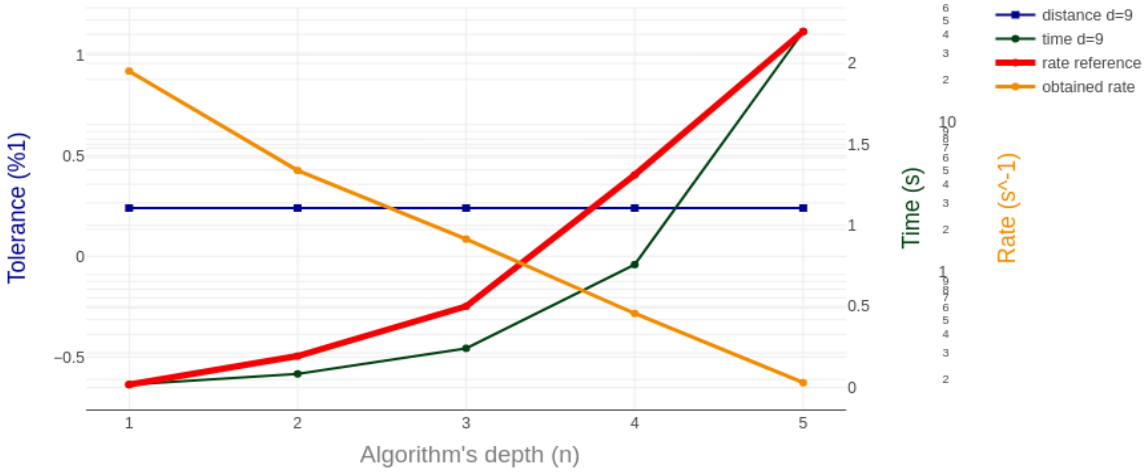

Figura 31: Resultados + Tasa frente a nivel de profundidad para buscador básico y subgrupo de 9 elementos